MVS Performance Management
OS/390 Edition

Stephen L. Samson
Senior Technical Staff Member
Candle Corporation

This is a re-issue in soft copy of the last hard copy edition published by McGraw-Hill "as is", as of the completion date at end of 1996. A few corrections and typographic changes have been made to improve readability. Blank pages inserted to ensure that chapters start on a right-hand page have been removed. Page numbers in the index and Table of Contents may be a bit off.

An update covering z/OS through Version 1, Release 3 will be made available in soft copy with more exploitation of hyperlinks and other features of the PDF format by approximately the end of 2001. A new edition covering the general availability and support of 64-bit virtual storage is planned for 2002.

Trademarks and registered trademarks used in this book are the property of their respective owners and are to be regarded as appearing with the appropriate ™ or ® symbols at their first mention. Predictions and evaluative statements are the opinion of the author only.
© Copyright 1997, 2001 Stephen L. Samson. Permission is granted to the recipient to make and print copies for personal use only. All other rights reserved.

This OS/390 Edition
is dedicated to my colleagues—at Candle, at IBM,
at SHARE and CMG, and elsewhere.
They have been generous with their input and advice,
and always constructive with their criticism.
I owe them much.

The ESA/390 edition's dedication:
This ESA/390 Edition
is dedicated with love
to my children,
Dara Michele Samson
and
David Scott Samson,
to my mother,
Stella Samson,
and most of all to my wife,
Wilma

The original edition's dedication:
To the memory of my father
William Samson
who showed me the pleasure of writing

Contents

PREFACE ix

1. MVS History and Evolution 1
1.1. Predecessor Systems 2
1.2. MVS As We Know It Today 7
1.3. Extrapolation 21
1.4. Chapter Questions 21

2. The Fall and Rise of MVS 22
2.1. Continued Evolution—More of the Same 22
2.2. What Went Wrong 23
2.3. A Recipe for Turnaround 27
2.4. Extrapolation 36
2.5. Summary 38
2.6. Chapter Questions 39

3. Physical Resources 40
3.1. CPU or Processor 40
3.2. Real Storage 53
3.3. Input/Output Resources 62
3.4. Summary 74
3.5. Chapter Questions 74

4. Virtual and Logical Resources 76
4.1. Virtual Resources 76
4.2. Physical backing of virtual resources 79
4.3. Logical Resources 87
4.4. Summary 98
4.5. Chapter Questions 98

5. Workloads and Service Levels **100**
5.1. Batch. 102
5.2. Started Tasks 115
5.3. TSO 121
5.4. APPC/MVS 126
5.5. OpenEdition MVS 129
5.6. Enclaves 129
5.7. Business Value and Service Level Agreements 130
5.8. Summary 130
5.9. Chapter Questions 131

6. MVS in SRM Compatibility Mode **132**
6.1. SRM—Using Part of the System to Manage Itself 133
6.2. Introduction to SRM Concepts 139
6.3. SRM Terms and Concepts 140
6.4. New Solutions in MVS/ESA SP 4.2 148
6.5. Changes in Version 5 Compatibility Mode 152
6.6. The Basic Flaw in the SRM Model 153
6.7. Summary 154
6.8. Chapter Questions 154

7. The Workload Manager in Goal Mode **156**
7.1. SRM and the Workload Management Problem. 156
7.2. Introducing the Workload Manager. 159
7.3. Creating the Service Definition 162
7.4. Measurement and Monitoring 178
7.5. Cautions, Limitations, and Extrapolations 178
7.6. Cheryl Watson's "Quickstart" Service Policy 182
7.7. Summary 185
7.8. Chapter Questions 185

8. Introduction to Performance Problems **187**
8.1. Initial Example 187
8.2. Types of Performance Problems 188
8.3. Resource Contention. 192
8.4. Special Problems of Client-Server Applications 195
8.5. Summary 195
8.6. Chapter Questions 196

9. Measurement and Prediction **197**
9.1. Approaches to Performance Management 198
9.2. Classifying Measurement Tools 202
9.3. One Kind of Modeling 214
9.4. Summary of Measurement and Modeling 234

9.5. Chapter Questions 234

10. Solving Performance Problems 236
10.1. Lost Opportunities 236
10.2. Configuring for Performance 242
10.3. Resource Exhaustion 258
10.4. Resource Contention 260
10.5. Summary 264
10.6. Chapter Questions 265

11. Application Performance Management 267
11.1. Open-Shop Workloads 267
11.2. Application Performance Vulnerabilities 268
11.3. Internal Inefficiency 270
11.4. Internal Contention 280
11.5. Why Applications Run Poorly 281
11.6. Application Design Choices 287
11.7. Ideas for Application Implementation 289
11.8. Considerations for Client-Server Applications 290
11.9. Practical Approaches to Cooperation 298
11.10. Summing Up 299
11.11. Chapter Questions 299

12. Summing Up and Looking Ahead 301
12.1. The MVS Journey—How Much Farther to Go? 301
12.2. The Ideal Large System 302
12.3. How Does MVS Meet the Ideal? 304
12.4. The Ultimate Wish List 304

Appendix A. SRM in Compatibility Mode—Concepts 306
A.1. Application to Workloads 306
A.2. Performance Groups 307
A.3. Service Intervals 307
A.4. Domains 309
A.5. MPL Adjustment 312
A.6. Storage Isolation 324
A.7. Dispatching Priorities 328
A.8. Summary 334
A.9. Questions 335

Appendix B. SRM (Compatibility Mode) Parameters 336
B.1. OPT Parameters 336
B.2. Installation Performance Specification (IPS) 369
B.3. Installation Control Specification (ICS) 383

B.4. Other SRM Controls 385
B.5. Other Controls 386
B.6. Comprehensive OPT, ICS, and IPS Example 386
B.7. Summary 393
B.8. Chapter Questions 393

Glossary **394**

Index **443**

PREFACE

Purpose of This Book

By the simple act of starting to read this book, you have expressed your interest in the idea that IBM's MVS[1] operating system is amenable to a discipline of performance management. Now that you're hooked, we'll attempt in these pages to define that discipline, to build on that interest and turn it into a solid commitment, and to describe and illustrate the means of implementing MVS performance management.

On the other hand, this book is neither an introduction to MVS structure and function[2] nor a treatise on MVS system programming. No knowledge of MVS internals is assumed or required for understanding. We deal with MVS's external controls and with the workloads found in most MVS installations. Some knowledge of application programming is useful.

What is Performance Management?

MVS is an operating system of great functional richness and correspondingly great complexity. Much of that complexity is devoted to routines that seek to manage the allocation of resources to work-

1 "IBM" is a registered trademark of the International Business Machines Corporation. "MVS" (the name is a trademark of the International Business Machines Corporation) was originally a short name for an IBM operating system, "OS/VS2: MVS," meaning "Operating System/Virtual Storage 2: Multiple Virtual Storage." MVS currently exists in several variants: MVS System Product Version 1 (MVS/370), MVS System Product Version 2 (MVS/XA), MVS System Product Versions 3 through 5 (MVS/ESA), and now OS/390. Except as noted, the content of this book applies in general to all recent and current MVS levels, although its principal focus is on MVS/ESA Version 5 and OS/390. Versions 1 and 2 are obsolete and no longer supported by IBM. The name OS/VS2 is rarely seen today.

2 Another book in this series, *MVS: Concepts and Facilities*, by Robert H. Johnson, meets the need for such a work exceptionally well.

loads according to the expressed desires of the installation's management. These desires are expressed to the system through numerous parameters that control parts of MVS's operation, and through decisions affecting the hardware configuration.

MVS performance management may be defined briefly as:

- acquiring the knowledge necessary to specify MVS control parameters and to make configuration choices
- translating service objectives into appropriate control parameters and a suitable physical configuration
- monitoring and measuring the system to verify compliance with service objectives and to assess the efficiency and effectiveness of the system
- dealing (with appropriate timeliness) with incidents of per formance that is inefficient, ineffective, or inadequate to meet service objectives

The definition will be refined and elaborated throughout the course of the text.

Audience

Those whose jobs include accountability for MVS performance are the primary audience for this book. Many others (the direct users of system services above all) are affected by the performance of the operating system and its workloads, and should derive some benefit from this book. Some of those include:

- operations production specialists
- operations managers
- "help desk" staff
- MIS application developers
- MIS managers
- MVS system programmers
- CICS system programmers
- IMS system programmers

- technical support managers
- managers of end-user organizations
- capacity planners
- resource usage accounting and chargeback staff

Overall Approach

It is easy to be put off by the sheer size and complexity of MVS and to conclude that the most effective management strategy is that of leaving the system "as received" until specific problems arise, and then solving them in turn. This book expresses a different approach: MVS responds well to informed planning, and installation policies can be implemented through MVS's control mechanisms. Our purpose is to demystify those mechanisms and show how MVS can become a well-behaved **managed** system. Adding to the motivation, MVS in its default form is simply not suitable for production use. Initial performance management actions are necessary to prevent the initial production experience from being a bad one. Even though the control of workload performance has been simplified in Version 5, the default settings (now in the form of a service definition) remain inappropriate for production use.

Irrational fear of MVS's complexity is to be avoided, but wary respect for that complexity is necessary. The redeeming quality of most of the actions recommended in this book is that, if done as recommended, they are reversible should problems arise. Those actions that require more care and planning are so identified.

Organization

In the first chapter, we examine the history of MVS, from predecessor systems through MVS/ESA (Enterprise Systems Architecture) Version 4.

Chapter 2 looks at the period of decline suffered by MVS in the early '90s and the significant initiatives IBM began with the April 1994 announcement. These new initiatives have restored MVS to its leading role. We'll examine the Version 5 and OS/390 changes in detail. Emphasis in these sections will be on developments important to control and performance management. The history ends with some speculation about the future of MVS.

Chapter 3 examines the physical resources managed by MVS. These include the physical elements of the computing system: CPUs, storage, and input/output (I/O) devices, with emphasis on direct access storage devices (DASD).

In Chapter 4, the actual working objects of MVS are considered. These are the *virtual* resources, which stand in for and rename physical resources, and the *logical* resources, which serve as the control focus for MVS's resource management functions.

We next consider the workloads managed by MVS. In Chapter 5, the workloads are classified and characterized according to their patterns of resource use and their performance management challenges. This chapter also introduces service level agreements, a key part of planning and managing service delivery.

Chapter 6 is an overview of the older central mechanism that controls how resources are allocated to units of work in MVS, the System Resources Manger, SRM. As of MVS/ESA Version 5, SRM in its old form is now known as *compatibility mode*. The details of compatibility mode operation may be found in Appendixes A and B.

The replacement for compatibility mode, *Workload Manager goal mode*, is described in detail in Chapter 7.

In Chapter 8, we examine performance problems, how they manifest themselves, and the kind of information that will be needed to expose such problems and their causes. The special performance problems encountered in client-server and other distributed application architectures are described and contrasted with those characteristic of more conventional systems.

We turn in Chapter 9 to a discussion of monitoring and measurement methods, considering the specific need for, and contribution of, each type of measurement technique in facilitating or simplifying performance management. Chapter 9 also includes the details of DASD modeling introduced in Chapter 3.

In Chapter 10, we apply all the foregoing material in covering the solution of performance problems, both chronic and acute. The emphasis in this chapter is on dealing with applications and subsystems, jobs and TSO users as they are, rather than as we wish them to be.

Chapter 11 pursues that ideal, considering the often controversial subject of application tuning. The specialized application of monitoring tools to this task can often be more effective in relieving system problems than the most concerted and protracted "system" tuning efforts.

Chapter 12 is an attempt to summarize the insights gained by an examination of the past and then to look beyond tomorrow's problems and extrapolate about ten years out into the future, with emphasis on the role of MVS or its successor, OS/390.

Appendixes A and B describe the controlling parameters of MVS in compatibility mode. In general, these influence the operation of SRM and appear in members of SYS1.PARMLIB. Other parameters controlling MVS's subsystems exist in other data sets, and some of the parameters having occasional use still hide in the dark recesses of control blocks and need to be adjusted when necessary through SUPERZAP or a similar mechanism. In these appendixes we suggest a set of standard parameters going beyond the IBM-supplied defaults. Some configuration guidelines are given in these chapters, and a comprehensive example of the parameter sets concludes Appendix B.

The book concludes with an extensive glossary of MVS and performance management terms. New terms are generally *italicized* at the point of first use. If a definition is not contextually obvious, it should be found in the glossary. Other stylistic conventions include:

- names of hardware instructions and their assembly language operation codes are shown in SMALL CAPITALS (LPSW or LOAD PROGRAM STATUS WORD)
- macro-instruction names are shown in *UPPER-CASE ITALICS* (*RESERVE* macro)
- system data objects and states are shown in UPPER CASE (RESERVE of a device, ENQ lockout)
- examples of card-image control statements are shown in **UPPER-CASE MONOSPACE SANS-SERIF BOLD**

Motivation

For four years or more, before I was initially approached by Jay Ranade to write a book on MVS performance management, I had speculated about such a possibility. The speculation began when I first became responsible for the MVS portion of Candle Corporation's old Fall Performance Seminars. It occurred to me that a solid 14 hours of lecture and supporting materials might form the nucleus of a book. In the intervening years, I wrote several series of MVS articles for the *Candle Computer Report* and observed the emergence of MVS/XA, the 3090 (and its follow-on models, later designated as Enterprise System/3090) and expanded storage, DB2, and the announcements of Systems Application Architecture and Enterprise Systems Architecture. The original edition of *MVS Performance Management* was the outcome of that time of readiness for such a book.

With the success of the original edition, it was natural to consider writing a next book, perhaps on a different subject. However, the evolution of MVS continued, bringing more change, in the September 1990 announcement of System/390 and Version 4 of MVS/SP (ESA/390), than in any prior announcement. Another huge set of announcements followed a year later, with more since that time. The ESA/390 edition of *MVS Performance Management* recognized that MVS changed dramatically, with new performance management problems and solutions, and radically different system structures.

Another need addressed in that edition was the requirement for chapter review questions, to accommodate the needs of educators who wish to use the book for teaching. I was flattered when I was asked for such additions, and I was happy to oblige them, as well as the many readers who provided helpful suggestions for additions or changes. There is not a "teacher's edition" with answers to the chapter questions. The questions are intended to have no "right answers" but rather to provoke investigation, reflection, and analysis. If successful, the questions will help the readers become activists for sound performance management.

As much as MVS evolved in the early '90s, there came a period of stagnation peaking in 1993, as IBM showed its first operating loss and the "mainframe is dead" hysteria became deafening. Fortunately, IBM was working on new initiatives during the bad times, and history was made again on April 6, 1994—one day short of the

thirtieth anniversary of the System/360 announcement. MVS Version 5 and all that came with it have reinvigorated MVS as the platform of choice for enterprise computing.

Since the changes were so great, it became evident that a third iteration of *MVS Performance Management* was needed. Other demands on my time and the desire to wait for a bit of experience with goal mode dictated the timing of this edition.

As MVS systems have become more affordable (OS/390 is now viable on a rather overgrown personal computer), the base of experienced MVS system programmers has been stretched thin to match the demand. System programmers' lives become more complicated as they respond to conflicting inputs about the care and feeding of MVS. Some say, "MVS is so fiendishly complicated that you should leave it alone .... You need special knowledge to change SRM parameters .... Trust us to guide you .... MVS defaults are OK ...."

Others (usually technical support managers) demand that the system be made to perform well as workloads change, and demand an explanation of each episode of poor performance. Certainly (they say) the knowledge necessary to manage MVS well, and to know why it's not well when it's not, should be sufficient to take on as simple a job as setting up a sound service policy.

Personal Biases and Matters of Style

MVS, its predecessors, and its successors are IBM-developed control programs intended to run on IBM systems. Several other manufacturers produce systems that are capable of supporting various levels of MVS, and manufacturers other than IBM have produced operating systems that are compatible with MVS in some respects. To the extent that the observations and advice in this book are valid for MVS and System/390 look-alikes, I am pleased to be of some assistance. However, my own experience and knowledge are focused on MVS and the systems for which it was developed. I make no claim of accuracy or insight for similar but not identical environments.

My own accuracy and insight are limited, in any event. All I promise is to discuss the lessons of my experience and to give advice that has worked in most cases. Neither I nor the publisher can in any way claim that the observations and advice contained in these pages are universally applicable. Before you

commit hardware, software, or human resources to implement any of the ideas presented in these pages, make sure that your situation matches that of the given example or scenario in all crucial respects, or that you have made due allowances for essential differences.

Writing style has received unusual attention in recent years, especially in one specific area: pronouns. To the greatest extent possible, I have tried to avoid the use of gender-specific pronouns. When the syntax gets too convoluted, I use the masculine form with its traditional generic connotation, at the risk of imperfect political correctness.

Trademarks and Registered Trademarks

This book could be cluttered with footnotes at the first mention of each term that is a trademark or registered trademark if each citation were to be given in full. Instead, I give full credit here to the commercial and intellectual property of the trademark holders whose products are mentioned in this book, and I apologize to those whose trademarks might have been inadvertently omitted.

IBM now routinely shows many trademark notices in announcements, presentations, and other materials relating to MVS and to other systems and environments as well. Many of the names now claimed as common-law trademarks may eventually become registered trademarks. Here is a reasonably complete list:

Registered trademarks of the International Business Machines Corporation: AIX®, IBM®, Image/Plus®, Micro Channel®, Multiple Virtual Storage/Enterprise Systems Architecture®, Multiple Virtual Storage/Extended Architecture®, NetView®, OS/2®, Personal System/2®, and PS/2®.

Trademarks of the International Business Machines Corporation: ACF/VTAM™, CICS™, CICS/ESA™, CICS/MVS™, CICS OS/2™, CICS/VSE™, Data Propagator™, DATABASE 2™, DB2™, DFSMS™, ECKD™, Enterprise Systems Architecture™, Enterprise Systems Architecture/370™, Enterprise Systems Architecture/390™, Enterprise Systems Connection Architecture™, Enterprise System/3090™, Enterprise System/9000™, ES/3090™, ES/3090-9000T™, ES/9000™, ESA/370™, ESA/390™, ESCON™, IMS/ESA™, Hardware Configuration Definition™, Hiperbatch™, Hiperspace™, MVS™, MVS/DFP™, MVS/ESA™,

MVS/SP™, MVS/System Product™, MVS/XA™, OS/390™, Processor Resource/Systems Manager™, PR/SM™, QMF™, SAA™, Sysplex™, System/360™, System/370™, System/390™, Systems Application Architecture™, SystemView™, S/390™, VM™, VM/ESA™, VSE/ESA™, VTAM™.

Candle Corporation's trademark product names are mentioned in the text. These trademarks include OMEGAMON®, EPILOG®, DEXAN®, and DELTAMON®. The logotype **!Candle** is also a registered trademark.

Boole and Babbage's RESOLVE and CMF; Computer Associates' CA-LOOK, CA-EXAMINE, CA-ACF2, and CA-TopSecret; Landmark's The Monitor for MVS (TMON/MVS); and SAS Institute's SAS are other commercial software packages mentioned in the text whose names are trademarks or registered trademarks of the respective vendors. Again, if any other trademark is not properly identified, the omission is unintended.

Acknowledgments

None of the events that led me to write this book and its predecessor editions would have been possible if Tom Apple and Hernan Reyes had not chosen me for a special assignment while I was at IBM in San Jose in 1980. I was privileged to work with Tom Beretvas and Ron Clark on a series of measurement experiments with what became both the Extended Swap Installed User Program (IUP) and the extended swap algorithm in MVS/SP 1.3.0.

My association with Tom Beretvas continued, even after my departure from IBM in 1983 and his, more recently. I've learned much from Tom in the specifics of MVS and even of VM, as well as in the discipline of performance analysis. Tom and I continue to engage in lively technical dialogue, especially on the few matters on which we had differing opinions. After that 1980 turning point, I became an MVS performance specialist, with emphasis on the paging subsystem.

I left IBM after 16 years to join Candle Corporation, the pre-eminent supplier of performance management software for the IBM mainframe environment. Tom Beretvas had a hand in that move as well, since it was Tom who introduced me to Marty Sprinzen, then the vice-president of Candle in charge of research and development. Marty brought me to Candle and gave

me a unique opportunity to split my time among technical support, R&D, teaching, and writing. This balance of tasks has continued through a series of assignments for nine years. It has kept me in touch with the "real world" as well as allowing me to influence Candle's future.

Candle started as a one-man company over 20 years ago. That man, Aubrey Chernick, still leads Candle in all ways. I greatly appreciate his support in this project, as well as that of all levels of Candle management.

Special thanks go to Cheryl Watson and Bernie Domanski, who took time from their *very* busy schedules to review my manuscript and contribute thoughtful and constructive suggestions for improvement. Although I have adopted many of their suggestions, I alone am responsible for the accuracy and correctness of this book.

Finally, a heartfelt word of appreciation to my beloved wife Wilma, who once again has supported this effort with loving encouragement mixed with uncharacteristic toleration for the (for me) inevitable mess attendant to the creative process.

Stephen L. Samson
Marina del Rey, California
December 1996

Chapter

1

MVS History and Evolution

A 30-year-old programmer may run jobs today under OS/390[1] that are older than he or she is. While MVS is a modern operating system—even state-of-the-art in many ways—it is a direct descendant of the first release of Operating System/360 in 1966. To a large extent, the problem program environment of MVS/ESA or OS/390 remains that of "OS."

We recount the history of MVS to help us understand its mechanisms and the biases implicit in them. We do this here not only to illustrate IBM's continual flow of improvement and innovation,[2] but also to note when and how constraints and limitations appeared in the MVS environment and how later changes overcame those difficulties.

The matter-of-fact appearance and subsequent relief of constraints is a natural consequence of the stair-step process of delivering improvements in hardware and in the many coordinated elements of supporting software. Yet, as many of MVS's shortcomings have appeared and been recognized by system programmers, a common reaction has been to institutionalize the deficiencies and the actions taken in response to them. In a

1 Since Release 1 of OS/390 (March 1996), "MVS" is just a synonym for the central operating system part of OS/390. However, in this edition we use the more familiar term except when explicitly speaking of OS/390.

2 IBM does not develop, test, and ship operating system improvements merely for the good of its customers. Major changes in MVS usually appear for IBM's strategic business purposes, such as to support new hardware products.

fast-moving field, this reaction can have the effect of unnecessarily perpetuating the effect of a temporary shortcoming. In this chapter, we attempt to identify such episodes of the birth of legends, or at least the lagging acceptance of good news.

1.1. Predecessor Systems

In this section, we look at the systems that preceded MVS, and the early releases of MVS that differed significantly from today's systems.

1.1.1. OS/360 MVT

MVT was the ultimate version of OS/360. The promises made when System/360 was announced on April 7, 1964 were kept in MVT to the extent that reality allowed. The original storage estimate—that a full-function multiprogramming operating system could exist in 64K bytes—seems ludicrous today. The original assertion—that work could compete freely for the real storage resource without the problem of fragmentation or the need for storage preallocation—never came to pass. MVT had to be modified drastically to accommodate two developments in the environment—multiprocessing and time sharing—and even the most devoted "OS bigot" would have to admit that MVT did not handle those developments very well.

Despite all of its growing pains, MVT supported the growth of the high-end System/360 and early System/370 hardware systems for over ten years and defined an application programming interface still supported in today's MVS. Indeed, a version of MVT to support IBM's 3031, 3032, and 3033 was released as late as 1978. The following concepts were introduced or made commercially successful in OS/360 including MVT:

- *Multiprogramming*[3] of independent work units without the need for fixed partitioning of storage
- *Multitasking*
- *Multiprocessing* using shared main storage, directed by a single control program
- Device-independent data management without the restrictions of a limited "file system"
- *Late binding* of resources to work units

3 See the Glossary for definitions of terms first shown in *italics*.

- I/O resource management across multiple users
- Main storage management across multiple work units
- Preemptive multiprogramming dispatching with *priority queuing*
- Dispatching priority adaptation to workload behavior
- *Time sharing* in a full-function operating system environment with full access to that environment

1.1.2. SVS

As unlikely as it might seem today, "virtual storage" was regarded as a radical development in the early 1970s. During the development of DOS/VS, the low-end operating system for System/370, a proposal for optional support of virtual storage was seriously considered. Those with greater vision prevailed, so no system called "DOS/almost-VS" was released.[4] Systems of intermediate size were supported by OS/VS1 (based on *MFT*). At the high end of the 370 line, the plan was two-pronged: A minimal extension to MVT would be released first, followed a year or two later by the system that the dynamic address translation hardware was meant for—a system to become known as MVS.

The original minimal extension to MVT was announced as OS/VS2 Release 1, or SVM, for "Single Virtual Memory." Soon after, the name was changed from SVM to SVS, for "Single Virtual Storage," when it became clear that the System/370 Models 155 and 165[5] were to be the last IBM systems with magnetic core *memory*, and that Models 158 and 168 signaled the changeover to semiconductor main *storage*.

SVS used virtual storage to solve (to a limited extent) the outstanding problem of MVT—storage fragmentation. Within the bounds of a single 16-megabyte address space, all SVS systems became equal in [virtual] storage size, if not in performance. Individual jobs ran in *regions* as in MVT, and the regions repre-

4 This curious ambivalence about making a full commitment to exploiting the power of the hardware platform can be seen today in the transitional operating systems for personal computers produced by Microsoft. Microsoft Windows and Windows 95 can be configured without virtual memory. The more powerful and "industrial strength" IBM OS/2 and Microsoft Windows NT have no such compromise; each has a mandatory "swap file."

5 The Models 155 and 165 were the last 370s built with magnetic core memories. They were supported by the VS operating systems only when equipped with an expensive hardware extension called the "DAT box." DAT is an acronym for Dynamic Address Translation, the essential element needed to support virtual storage.

sented real boundaries and constraints—the subdividing of a still-precious span of storage to accommodate a balanced workload such that all fit without waste. In SVS, *TSO* still ran in multiuser regions, and each session's storage was wholly swapped out and in at the start and end of each terminal wait. Little if any *overcommitment* of real storage was attempted.

1.1.3. OS/VS2 Release 2, 3, ..., 3.6, 3.7, ...

The early releases of MVS from 1973 to 1975 were, frankly, exciting. MVS was described at that time much as MVS/XA was later described in its infancy, as a "special" system for very large configurations needing its special capabilities. With the immediate need for support of virtual storage covered by SVS, the evolution to MVS could be allowed to proceed with cooperation and feedback between IBM and the first MVS users. (When IBM cured MVS's early growing pains, the restraint gave way to a determined selling effort, but the early words of caution had had their effect—SVS survived far too long.) During those early years of MVS, a number of constraints began to appear. Although the constraints eventually were understood and overcome, they had an influence in the MVS environment long after they were no longer current problems, and the descriptions of the constraints became generalized and divorced from their origins. These pervasive "legends" included:

- **"Paging 'just happens' in MVS."** Global *least-recently-used* (LRU) management of real storage replacement was used in SVS and the first MVS release. Consequently, *demand paging* occurred everywhere in the system, limiting the ability to exploit fast CPUs given the limited amount of very expensive real storage in the early MVS machines. This problem was ameliorated with the availability of *storage isolation* in MVS/SE2, and with the dramatic increase in basic and optional real storage sizes available for more recent systems. However, there remains a persistent belief that paging is an uncontrollable phenomenon in MVS, and workloads are often managed as if real storage is severely limited, whether it is or not. Mismanagement of paging had another expression: Systems were not fully loaded to avoid incurring excessive paging delay, instead of benefitting from simple *SRM* controls that managed the paging rate in a fully loaded system.

- **"TSO swapping is not handled well in MVS."** In early MVS, the pages of inactive TSO users were left in real storage, subject to ordinary page stealing (as noted above). It soon became clear that LRU page stealing was too inefficient to handle the reassignment of large numbers of idle frames. TSO *swapping* was introduced in one of the early variants of VS2 Release 3. The hardware and software approaches to swapping and their controlling algorithms have been one of the most frequently changed (and usually improved) areas of MVS over the years.

- **"TSO and other interactive workloads don't work well together in MVS."** The utilization of page data sets, caused by TSO swapping and the large number of *page faults* resulting from rigorous *swap trim*, was so high that paging response time for non-TSO workloads often became unacceptable. Even though the numerous changes in TSO swapping algorithms over the years yielded significant relief as early as 1981, it has remained an enduring legend in MVS that TSO and other subsystems, such as CICS or IMS, don't coexist well. Solving the mixed-workload problem was an essential prerequisite to the effective use of large configurations, and the means of solving the problem has been available for years.

- **"MVS needs too much real storage."** The original real storage estimates for MVS were far too optimistic; the 16-megabyte limit on real storage was in jeopardy before a new technology replacement for System/370 would become available.[6] A jury-rigged solution became available when the real storage limit was extended by a factor of 4 in late 3033s and the original (System/370) version of the 3081. What appeared to be a long-term solution came with the MVS/XA announcement of 31-bit (2-gigabyte) real and virtual addressing. By 1989, however, the 2-gigabyte virtual storage limit was routinely touched, the 2-gigabyte real storage limit was only 2 bits away, and the need for increased real storage was being met, albeit with uneven success, by the use of expanded storage. This particular legend may remain viable for some time whenever MVS's use of storage is measured against that of systems with less comprehensive architectures.

6 It's difficult to maintain perspective in these matters. Personal computers are now often configured with 32 or more megabytes of main storage; desktop (UNIX) work stations support 128 megabytes or more.

In discussing these legends, we've had to jump around MVS's history a bit. We return now to a more orderly review of successive major changes in MVS.

1.1.4. System Extensions (MVS/SE)

In March of 1978, MVS came to a turning point. The first implementation of MVS System Extensions (MVS/SE1) was announced. Nine years earlier, IBM had announced the "unbundling" of hardware and software. Control programs required for the operation of IBM systems were classified as "system control programming" (SCP) and were provided at no charge.

MVS/SE became the first IBM operating system for which a fee was charged, and the first to require hardware features beyond those in the System/370 base instruction set. Beginning with SE1, an increasing number of MVS software algorithms have been placed in microcode, leading both to increased efficiency and to a "moving target" for other manufacturers attempting to duplicate the MVS hardware environment.

The IBM 3033 with System Extensions was the first generally available IBM system to benefit from an approach to system design based on the active consideration of operating system needs and optimization opportunities. (A less rigorous hardware-software design approach, seemingly with some isolation between engineers and programmers, had been characteristic of System/360.) An additional benefit to IBM was the beginning of what has since become substantial annual revenue from the licensing of system software products.

The success of the 3033 with MVS/SE1 led to two MVS software developments, one a dead end for the no-fee MVS System Control Program, and the other the forerunner of the licensed-for-a-fee succession of MVS offerings that continues today.

The evolution of "free" MVS stopped with Release 3.8 in 1979. Although MVS 3.8 is thoroughly out of date today, it is in the public domain and still in circulation. It is used to support hardware configurations not capable of exploiting the microcode-assisted features of current levels of MVS, and continued to be the "base," at least theoretically, over which MVS releases through Version 3 were installed. Thus it is that an *RMF* report for an MVS/SP Version 3 release still said "3.8" in its heading. The announcement of

3.8 made it clear that there was to be no future functional enhancement of a no-fee MVS.

MVS 3.8 was finally laid to rest with the availability of MVS/ESA SP Version 4. MVS/ESA beginning with Version 4 is not built on a 3.8 base.

The other development that followed the success of MVS/SE1 was the announcement of SE2 in August 1979. In addition to more microcode assistance, SE2 introduced most of MVS's SRM controls for storage isolation, dispatching priorities, and domains' contention indexes. Those controls survived virtually unchanged until the release of MVS/ESA SP Version 4.2.

1.2. MVS As We Know It Today

MVS/SE2 set the pattern for MVS as it exists today. Successive MVS releases have followed in the same mold, with growing dependency on microcode.

1.2.1. MVS System Product Version 1 (MVS/SP)

MVS/SP1

On June 11, 1980, MVS packaging, nomenclature, and direction were changed once again. Two releases of a new kind of software called "MVS System Product" (MVS/SP) were announced. What was then called SP1 was functionally little more than a repackaging of SE2. As part of the repackaging, the complex "SU" (selectable units) scheme for MVS distribution was abandoned. MVS distribution with SUs had begun in 1976, when real storage constraint was at its worst, and was based on the idea that "optional" parts of the MVS nucleus could be left out when not needed, thus saving fixed virtual storage and thereby real storage. Eventually, some priced enhancements to "free" MVS were shipped as SUs; unpriced SUs continued for such additions as device support.

SP1 provided support for a significant assortment of new hardware, including:

- the 3033 Extensions feature, providing hardware support for cross-memory services, making possible the "horizontal expansion" of the MVS control program with acceptable performance
- 3375 and 3380 direct access storage devices (DASD)

- Data Streaming Channels to support the new, higher data transfer rates of the 3375 and 3380, as well as the Speed Matching Buffer for support of the new devices on processors whose channels could not be (or were not) upgraded

MVS/SP2

The second of the two releases announced in June 1980 was MVS/System Product Release 2, MVS/SP2. SP2 exploited the cross-memory addressing mechanism introduced in SP1 by introducing the notion of "horizontal growth" in MVS. As the need grew for enhanced function and durability in the MVS environment, more operating system code became necessary. At the same time, the encroachment of nucleus and common area code and data on each address space's private area became increasingly unacceptable.

Cross-memory services made it possible, with insignificant overhead, to pare off portions of the operating system and place them in their own independent address spaces. Code and associated data could be removed from the commonly addressed area, thus relieving the constraint on virtual storage addressability in the private area. A section on virtual storage constraint relief (VSCR) became a common feature of IBM announcements, dating from the SP2 announcement and continuing through the latest MVS/ESA announcements. It is also possible that when IBM changed global data areas into private areas, a change in keeping with current "software engineering" notions, MVS became somewhat less complex in its potential data flows, if more complex in structure.

Global Resource Serialization (GRS). The first services to benefit from this restructuring technique were the enqueue/dequeue (ENQ/DEQ) routines and console services. The ENQ/DEQ restructure made it possible for IBM to address a problem of long standing. Multisystem configurations connected through shared DASD suffered from unpredictable and often severe degradation caused by ENQ's use of the volume RESERVE function to ensure integrity of data sets shared across systems.

Although some non-IBM solutions to this problem were available, they relied on a designated shared device to maintain global ENQ data. IBM's planners and developers were motivated to provide an all-IBM solution to the global data-sharing problem, substituting the potentially higher performance of channel-to-channel adapters

for the conspicuously shared device. Because of the encroachment of IBM-supplied code on the users' code and data area (the "private area") and the potentially large size of a global enqueue manager, the solution had to wait until cross-memory services became available in MVS/SP2.

The new service and its address space were known as GRS, for Global Resource Serialization. Even in non-shared systems, all ENQ and DEQ requests, as well as all related control blocks and data areas, are respectively executed through, and resident in, the GRS address space. In shared DASD systems, RESERVEs may be "demoted" to a new class of ENQ with a scope of SYSTEMS. Data-sharing information is passed, along with a token denoting control, among systems connected by a "ring" of channel-to-channel (CTC) adapters.[7]

The CTC ring introduced some new potential performance problems. In the original SP2 implementation, restarting a broken ring was very difficult. Instead of reducing shared DASD delay, SP2 GRS could increase it, either when the ring was broken or when a slower system held the token for an excessively long time. The latter source of delay was not to be overcome until MVS/ESA Version 4 became available.

Some non-IBM global enqueue managers employ a "star" configuration instead of a ring. If the shared device is cached DASD or a solid-state device, the performance of such a subsystem can be comparable to that of global GRS at its best, and less susceptible than GRS to delay caused by a software problem in a single system. With the availability of the *coupling facility* as a suitably fast shared device, IBM adopted the star configuration for GRS in a *parallel sysplex* as an alternative to the ring in OS/390 Release 2.

CONSOLE. Another service address space introduced in SP2 was CONSOLE. CONSOLE provided the base for a continuing series of enhancements to the handling of the operator message stream, thus dealing with another key impediment to MVS's ongoing growth. Efficient processing of the console message stream is a prerequisite for automating the console functions and for eventually attempting unattended MVS system operation, a slowly emerging trend in large data centers.

7 It might be that the "pass-the-buck" ring design of GRS was the first IBM token-ring network. It might also be that *JES3* did it years earlier.

Circumstances (the announcement of SP3 in November of 1980 and the short time span between the availability dates for SP2 and SP3) made SP2 something of a shadow release. Very few customers installed it, preferring to wait for the more obvious benefits of SP3. This development worked to the benefit of both IBM and its customers, since the initial GRS implementation's fragility and lack of recoverability were unacceptable to many MVS customers. As happened in many instances of legend formation, GRS continues to have a somewhat tarnished reputation in spite of continual rounds of improvement.

MVS/SP3

November of 1980 brought another crop of major advances in the world of MVS. The major hardware announcement was the first 3081, later redesignated as the 3081D. To support the new capabilities of the 3081, as well as those of the 3033s (which had been further enhanced with a new feature called 3033 Extensions), a new release of MVS was announced, known at that time as MVS/SP Release 3, or SP3.

SP3 (still installed "over" MVS 3.8) brought many far-reaching changes. A year later, after MVS/XA was announced, it became clear that some of these new capabilities were in the nature of "positioning" for XA, but SP3 also became the definitive surviving version of MVS/370. The new functions and support in SP3 included:

- 3081 support
- 3033 Extensions support, highlights of which (and of the 3081) included *queuing channels* and the I/O *suspend/resume* feature, which provided the basis for *seldom-ending channel programs* (SECP). Both of these hardware extensions facilitated the offload of high-priority processor cycles to the I/O subsystem, foreshadowing the much more significant I/O overhead reduction that accompanied Extended Architecture and MVS/XA. Another element of the new hardware was very efficient support of cross-memory services.
- Extended swap. As MVS evolved, TSO swapping and its management were revised often. The original VS2-3 version imposed unacceptable loads on the paging subsystem as the pages held by idle address spaces were stolen. Physical swapping made management of the swap paging load somewhat

more efficient, and the introduction of the swap data set (at first only a single one on most systems) brought a modest amount of page data set relief. As real storage sizes began to increase, the introduction of logical swap in SE1 unloaded the page (and swap) data sets still more, but only when the real storage resource was already underutilized.

With all of these changes, it was still not wise to mix a TSO workload with a multiuser online system, such as CICS, on the same MVS system, because the appearance of real storage constraint led to the joint degeneration of both TSO and CICS response time. Single-task subsystems such as CICS are much more profoundly affected by paging delay than TSO, so even storage isolation was not very effective in shielding interactive subsystems from the long response time delays associated with heavily loaded page data sets. A new approach to solving the paging problem was needed.

Other manufacturers attempted to deal with paging performance problems by using fast solid-state paging devices. These devices first appeared to the system as IBM 2305-2 fixed-head files (also known as "drums" because they performed very much like genuine drums with a single cylindrical recording surface and one head per track). IBM had stopped making 2305s, and the solid-state devices (SSDs) filled the gap with better performance and reliability. Eventually, the storage capacity increased, the simulated device became the IBM 3380, and some provisions were made for nonvolatile storage. In light of subsequent developments, it becomes clear that any long-term success of the SSD approach would have inhibited migration to the 3090s with expanded storage.

IBM's paging solution was an algorithmic invention. The nature of TSO swapping was changed, making the speed of mass data transfer much more significant than that of fast single-page retrieval. Before extended swap, the set of pages swapped out did not match the swap-in group, or *swap group*. Changed (but not necessarily recently referenced) pages were written out, while recently referenced (but not necessarily changed) pages were swapped in. The profound invention in extended swap was to recognize that a swap group with the same population for swap-in as for swap-out could be moved intact via blocked I/O. Even though more pages were moved, the blocking reduced

the overall swap-in time. The introduction of extended swap had several benefits to IBM:

- The local page data sets could be significantly unloaded by moving swap groups to and from swap data sets. This in turn meant that demand paging response time would improve, facilitating the efficient combining of workloads on larger systems.
- The swap data sets could be placed on 3380s with their 3-megabytes-per-second data transfer rate. Solid-state devices provided little added benefit in this application because data transfer time (rather than page-finding time) for the first time dominated the swap-in delay. The 3380 offered a faster data transfer rate than the first-generation solid-state devices.
- Use of storage isolation along with extended swap eliminated more than half of TSO page faults and thus made consistent sub-second response time possible for TSO, leading to improved user productivity. Storage-constrained systems could then achieve higher levels of CPU utilization, as an inhibition on servicing latent demand was eased.
- In systems without swap data sets, *contiguous-slot allocation*, a new approach to constructing paging channel programs, brought "blocked"[8] paging I/O to local page data sets. Since most of the device-busy time on the paging devices was related to per-page overhead, contiguous-slot made the benefits of extended swap available on systems with "locals-only" paging configurations. Because the size of a block on a 3380 was larger than the swap set size of 12 slots, efficiency was potentially higher than with swap data sets. Two factors held back the full adoption of locals-only: the need for storage isolation of TSO along with locals-only was not appreciated at first by IBM's advice-givers, and the initial implementation of contiguous-slot was flawed.
- Most important for IBM's large systems strategy, the value of "drums" or their solid-state replacements was largely negated in the TSO environment.

8 Contiguous-slot I/O is not blocked in the sense of supporting multiple logical records per physical record, but uses command chaining of successive READs or WRITEs within the same cylinder to produce a similar level of efficiency.

- A special form of Virtual I/O (VIO) known as Virtual Fetch was implemented as a set of MVS primitives and used in a new release of IMS. Virtual Fetch helped reduce IMS delays due to repeated loading of nonreenterable[9] programs.
- Although it was introduced in SP1, extended real addressing above 16 megabytes was not well used before SP3 became available. As SP3 could support many more TSO users and more VIO than prior releases, the added storage was used as a reservoir of pageable frames, thus avoiding paging I/O. Real storage sizes through 64 megabytes were eventually supported in SP3, although little benefit could be derived from real storage sizes in excess of 32 megabytes.

  SP3 was the last major release of MVS/370, although it eventually acquired a different name. Seven modification levels of SP3 were shipped, providing new device support and new releases of the two alternative MVS job entry subsystems, JES2 and JES3. MVS/370 was withdrawn from marketing on December 31, 1991, with withdrawal of support scheduled for a year later.

1.2.2. MVS System Product Version 2 (MVS/XA)

MVS/SP 2.1.0 and MVS/370 Redesignation

Less than a year after the SP3 announcement, IBM dropped an even bigger shoe. The 3081 was revealed as a machine with a dual personality. In addition to the System/370 instruction set (augmented with a great deal of specialized MVS *microcode*), the 3081 also had another mode of operation: System/370 Extended Architecture, "XA."

For years it had been apparent to IBM's product and marketing planners that System/370 and MVS on System/370 had architectural limits that could no longer be tolerated. The limits included only 16 megabytes of virtual addressability, no more than 30 high-speed channels, central processor complexes with no more than two CPUs,[10] real storage limited to 16, and even-

9 The term "reentrant" is sometimes used as a synonym for "reenterable." "Reentrant," and related barbarisms such as "reentrancy," should be avoided, regardless of common usage, since "reentrant" (a word whose only merit is a smaller number of syllables) has the plain meaning of "a program that re-enters itself"—in other words, a loop! The essence of the correct term, *reenterability*, is the suffix *-ility*, correctly suggesting the ability to be re-entered or reactivated while a previous activation had not yet concluded.

tually 64, megabytes, and an I/O subsystem that made continual disruptive demands on the CPU and the operating system to drive I/O and deal with the ensuing interrupts. Carryovers from System/360 defined many of the 370's limitations. A new system architecture was needed, but the customers dependent on the System/370 and its operating systems could not be expected to convert their applications to a new hardware and operating system architecture.

From its inception, XA had two principal objectives: to overcome the architectural limitations of System/370, and yet to allow the vast majority of System/370 programs to run without change. MVS/SP 2.1.0 accomplished both these purposes, but brought some often-confusing retroactive changes in nomenclature. The new operating system was called "MVS System Product Version 2." The prior version was obviously Version 1, although it had never been called that before. SP3 became known as SP1.3; its third modification level was, for example, SP1.3.2. (Versions and releases are numbered from **1**; modification levels begin at **0**.)

SP2.1.0 was functionally equivalent to SP1.3.0. As part of establishing the change of direction attendant to the release of a new operating system, the operating system was split into two corequisite parts. SP2.1.0 proper was only the Base Control Program, or BCP. The collection of "Data Facility" products, developed in IBM's Santa Teresa Laboratory in San Jose, California, was gathered together as one product and merged with the formerly "free" data management portion of the operating system and sold as Data Facility Product/XA, or DFP/XA. A DFP/370 was announced shortly thereafter. The BCP/DFP structure became standard in MVS, although nomenclature changed once again when the basic DFP functions were later repackaged as DFSMSdfp. IBM later followed the same pattern in creating another broad-function licensed program: several network control, debugging, and measurement products were gathered together as the first version of NetView.

In most cases the conversion to MVS/XA appeared to be less disruptive than the move from SP1 to SP3. The easy conversion in

10 Although the architecture of MVS/370 could theoretically have supported up to 16 processors, no System/370 processor complex exceeded a two-way configuration because of engineering constraints in the basic designs. MVS/370's design allowed for only two-way MP. Could *n*-way MP have been made to work on MVS/370? Many operating system changes would have been necessary, and the testing effort alone could have made a lifetime career.

terms of system programmer effort was often followed by a nasty shock when the production workload was moved to the new XA system. The surprise was reminiscent of the one experienced when an SVS system was converted to MVS. XA is built on a much larger scale than that of MVS/370, similar to the relationship of MVS to SVS. A workload requiring 16 megabytes of real storage on MVS/370 might need 18–20 megabytes under XA. Storage increment sizes on the first 3081s were 8 MB, so an upgrade to 24 MB was required. The real-storage shock wore off in time as later machines were announced with larger minimum and maximum sizes, at a smaller cost increment per megabyte.

Subsequent SP2.1 Releases

Following at regular intervals after SP2.1.0 came several additional XA announcements:

- SP2.1.1 added support for cache DASD controllers.
- SP2.1.2 brought a revised Auxiliary Storage Manager that removed inefficiencies introduced with SP1.3.0. The same changes were also made available for retrofit to MVS/370, SP1.3.1 and above.
- SP2.1.3 added support for the 3090-200 and expanded storage. Interim packages (an echo of SUs) added to the SP2.1.3 base supported vector processors on the 3090s and a set of availability enhancements, on a mutually exclusive basis.
- SP2.1.5 added support for the 3090-400.
- SP2.1.7 integrated the interim packages based on SP2.1.3 and introduced significant dispatcher improvements.

MVS/SP2.2.0 and 2.2.1

MVS/XA evolution continued with two releases of "SP2.2": SP2.2.0 and SP2.2.1. They differed only in that 2.2.0 had a new JES2 component, while 2.2.1 included a new level of JES3. Changes introduced in the 2.2 level of XA include a vastly simplified system customization procedure, removal of miscellaneous functional constraints dating back to the first release of OS/360, and a new kind of data object called Data in Virtual (DIV). DIV has the potential of providing efficient support for very large data base applications with sparse reference patterns. DIV is supported by a new kind of

VSAM data set called a Linear Data Set, which is also used directly by IBM's relational data base subsystem, DB2. The VSAM Linear Data set support is in the Data Facility Product (DFP) and the DIV function built on it is in the Base Control Program.

MVS/SP2.2.3

On June 20, 1989, IBM announced the last release of MVS/XA, JES2 and JES3 versions of MVS/SP2.2.3. These releases were equivalent to SP2.2, except that they operated with the Data Facility Product of MVS/ESA, DFP 3.1, and a subset of its system-managed storage features. The purpose of these final releases was to allow installations with an inventory of ESA-capable systems to migrate to MVS/ESA while allowing older XA-only systems to coexist in the same data center.

MVS/XA was eventually dropped from IBM's product line, and MVS Version 2 service was discontinued on September 30, 1995.

1.2.3. Enterprise Systems Architecture and MVS/ESA

On February 16, 1988, IBM announced a series of far-reaching extensions to MVS. A new hardware architecture, Enterprise Systems Architecture (ESA), was announced, along with a new version (Version 3) of the MVS System Product. The 3090 family of processors was redesignated the ES/3090; new top models of the 4381 (90E, 91E, and 92E) became compatible with ESA and were announced as ES/4381s.

ESA is an evolutionary extension of XA. New extensions to data addressing introduce the idea of *data spaces* which are addressed using an additional kind of register and object-oriented *tokens*. Data spaces are data-only counterparts of address spaces that enable application programs and subsystems to address essentially unlimited amounts of data. The introduction of data spaces in ESA seems to represent the end of at least one aspect of Von Neumann architecture in the systems that originated with System/360. One basic characteristic of a Von Neumann architecture (named for John Von Neumann, an early pioneer of computer designs) is the equivalence of programs and data. A program could branch to any arbitrary address subject to boundary constraints, and the hardware would dutifully attempt to execute the bit strings found there as "instructions." Such potentially harmful operation is not possible when data is segregated from code in a data space.

MVS/ESA also includes the notion of *hiperspaces* (high performance [data] spaces) that can substitute an area of expanded or

auxiliary storage for a data set in pursuit of a prime ESA objective: avoiding input/output activity on the critical performance path.

Version 3 of MVS/SP went through a series of releases ending with SP 3.1.3. The second release, with the curious designation of 3.1.0e (that's "little e"), introduced the Version 3 Data Facility Product, now termed DFP/ESA, along with the delivery of hiperspace support.

DFP/ESA brought a major change in data management, Data Facility Storage Management System, or DFSMS. Until then, little had changed since OS/360 Release 1 in the way data sets were allocated on physical devices. For direct access storage devices (DASD) especially, knowledge of physical device characteristics was needed to ensure efficient use of space and satisfactory performance. This knowledge was required of each TSO user and every application programmer coding Job Control Language statements. With system-managed storage (SMS), the usual name for DFSMS, the need for special knowledge is restricted to one or a few *storage administrator(s)* whose task it is to create a series of *class definitions* for data sets. These classes are then in turn used (usually implicitly) by those creating data sets; the information relating to specific devices or physical characteristics can be omitted. DFSMS through its class structure can handle such matters as retention and archiving, assigning selected data sets to high-performance or high-integrity devices, and controlled use of device pools according to installation policy.

A "toleration mode" of DFSMS was retrofitted to DFP/XA so that migration to ESA would not be hindered.

IBM announced significant extensions to MVS/ESA and ES/3090s in October of 1989. New "J" and "JH" models of the 3090 featured reduced cycle times and larger maximum storage sizes. PR/SM[11] in the new models supported VM/XA in a logical partition, as well as a rudimentary form of dynamic storage reconfiguration.

In MVS/ESA, new facilities were announced for I/O avoidance, especially in heavy production batch environments. A new function called Hiperbatch™ could place sequential and VSAM data sets in hiperspaces to reduce I/O delay. The new I/O avoidance measures were available in general without the need to rewrite programs or even to restructure complex *JCL* streams. A hardware feature called Move Page is a prerequisite for Hiperbatch, first supported in the operating system by an enhancement to MVS/ESA Version 3.

11 PR/SM (Processor Resource/Systems Manager, pronounced "prism") is discussed in detail in Chapters 3 and 4.

1.2.4. System/390, ESA/390, and MVS/SP Version 4

IBM further increased the pace of change on September 5, 1990. In a huge announcement package, IBM announced (among many other new products) a new line of processors, a new processor architecture, a new framework for system management, significant new connectivity functions and options, and new versions of each principal *mainframe* operating system, including the first two releases of Version 4 of MVS.

System/390 and ES/9000

The new architecture, ESA/390, was embodied in three new series of processors, collectively designated as ES/9000. The three processor families were somewhat uneven in new features and encompassed a performance range of over 100 to 1, with processor instruction execution rates ranging from less than 2 MIPS[12] to over 220 MIPS.

All of the new processors could be (optionally) configured to include the new Enterprise System CONnection (ESCON) channels.

The most homogeneous of the three families was the middle one; these were air-cooled mainframes similar in overall configuration to the IBM 4381 processors that they replaced. These "air-frame" IBM 9121 processors were vastly more powerful than their predecessors.

The rack-mounted entry-level machines (denoted 9221s) included updated 9370s and an all-new Model 170 with a deferred delivery date. Many of the new features (such as the ESA/390 instruction set) were also deferred.

The high-end water-cooled machines (9021s) fell into two sub-categories. The first ones available mirrored the ES/3090J series in internal design and relative power, from the 330 and 340 uniprocessors up to the four-way 620 and the six-way 720. Shorter cycle times provided a modest performance improvement over similarly configured Js. No five-way was announced.

The second group of top models initially announced included the 820 four-way and the 900 six-way. (An 860 five-way was announced in April 1991.) These three models were not warmed-over 3090Js; they were of a new design with significantly improved concurrency and bandwidth, but

12 MIPS is an acronym for "million[s of] instructions per second," a measure of processor power that is deservedly derided by everyone, who nonetheless all end up using it under protest. Variables that can affect the accuracy of MIPS rate comparisons include the architectural approach to the processor design (as in RISC vs. CISC), the degree of parallelism built into the processor, the workload mix used or modeled to make the comparison, the I/O capacity and bandwidth of the processor complex, and the design, hierarchy, and bandwidth of processor storage.

their initial delivery dates were a year off. We'll refer sometimes to these models as "advanced-design" 9021s, later designated by IBM as the "520-based" 9021s. (The 520 was the uniprocessor version of this generation of systems, announced some time after the larger models.)

SP 4.1

Following a familiar pattern, IBM limited the new content of the first release of Version 4. Much was new in SP 4.1, but hardly enough in itself to merit the designation as a new version. However, the introduction of ESCON channels is the type of hardware innovation that has always been accompanied by an MVS version change. High-speed serial channels, implemented with glass fiber cable connections, can revolutionize data center designs and make possible the connection of multiple MVS systems in a new aggregate called a *sysplex* (SYStem comPLEX). Indeed, the combination of the hardware change and the full set of Version 4 software changes in SP 4.1 and SP 4.2 constituted a new generation of IBM mainframes.

MVS/ESA SP 4.1 was a transitional release. It provided the new base of Version 4, exorcizing the ghost of MVS 3.8. It also introduced a new console subsystem as a prerequisite to the full implementation of sysplex. A new inter-system communication protocol based on ESCON was included, making possible improvements in GRS and timer synchronization among the individual systems of a sysplex. A final new feature was a Hardware Configuration Definition (HCD) process superseding the MVSCP process introduced in SP 2.2.

1.2.5. MVS/ESA SP4.2

MVS/ESA SP4.2 brought a larger number of significant changes than any previous MVS release. A brief list included:

- **APPC/MVS** for cooperative or distributed applications and subsystems.

  This new type of address space and communication mechanism was used at initial availability by JES, CICS, IMS, and DB2. There is a strong indication that more IBM subsystems will trade old structures and connectivities for APPC/MVS as future releases appear.

- **Dynamic Reconfiguration Management** built on the Hardware Configuration Definition introduced in SP 4.1.

(A "wishlist" favorite dating back to OS/360 Release 1 was the ability to change device configurations without an IPL.)

- **PR/SM Automatic Reconfiguration Support**, by means of which an MVS system in a "standby" partition can take over the storage and devices of a failing production partition, given the appropriate programming support to "listen" to the "heartbeat" of the production partition.
- **Console Integration** completed a line of development begun in SP 4.1, by using the hardware console for MVS initialization and thereby eliminating the need for a dedicated local terminal control unit. The operating console could be a TSO/E terminal.
- **Numerous SRM changes** were implemented:
  - Many parameters were removed or changed.
  - Parameters were introduced to permit better control of access to expanded storage.
  - New algorithms were implemented to improve real storage control for "problem" (numerically intensive) workloads and others.
- DFP 3.3 support of Dynamic Reconfiguration Management provided the link from device reconfiguration to use of that updated information in making allocation choices.

1.2.6. ESA/390, 1991 and 1992 announcements

Just a year after the massive September 1990 announcement, IBM again produced a (slightly smaller) September Spectacular. The principal thrust of the announcement was hardware; seven new processors were announced along with a triple-density 3390 Model 3 and a new intermediate DASD subsystem, the 9340. Also announced were increased connection distances for all ESCON channels and increased data transfer rates (from 10 megabytes per second to 17 megabytes per second) on (only) the 520-based 9021s. The previous three models of that design (9021-820 four-way MP, 9021-860 five-way MP, and 9021-900 six-way MP) were augmented by a full selection of additional processor configurations: uniprocessor, dyadic, triadic, and a two-way multiprocessor. Added to the 9121 "air-frame" family were three new multiprocessors: a two-way, a three-way, and a four-way.

All of the ES/9000s from the Model 170 on up supported a new hardware feature called Subsystem Storage Protection, as well as several less significant enhancements. Subsystem Storage Protection provides the equivalent of individual storage protection keys for individual programs within a subsystem address space. The feature was initially supported in CICS/ESA.

New hardware pushed out the old, as the ES/3090 E and S models and all 3380 models were withdrawn from marketing.

MVS/ESA was enhanced for the new announcement, although the MVS announcement, of MVS/ESA SP 4.2.2, was made several weeks earlier. SP 4.2.2 is required for support of Subsystem Storage Protection, as well as enhancements in programming and operations support of APPC/MVS.

An announcement in February 1992 introduced two more air-cooled ES/9000s: a low-entry air-cooled frame model, the 9121-190, and the first rack-mounted model with more than one processor, the 9221-200. Like the models introduced earlier, the 190 and 200 were fully capable MVS/ESA systems, especially when teamed with the IBM 9340 DASD subsystem.

1.3. Extrapolation

The story of MVS evolution through Version 5 and beyond continues in the next chapter.

1.4. Chapter Questions

1. List five strengths and five weaknesses of MVS that became evident in mid-1993.
2. Concentrate on the weaknesses. State how each of these weaknesses might have tended to make installations abandon MVS.
3. How might each of these weaknesses have been overcome by changes in MVS? If any of your weaknesses has no cure, set it aside as an *essential* weakness of MVS.
4. For the strengths you listed in question 1, how may each be made even more beneficial to the acceptance and long-term growth of MVS?
5. How can effective performance management increase the strengths and mitigate the weaknesses of MVS?

Chapter

2

The Fall and Rise of MVS

MVS fell on hard times in the early 1990s. In fact, the very viability of MVS and the platform on which it ran became questionable. How did IBM's cash cow become diseased, and how did it come back from the brink of death?

2.1. Continued Evolution—More of the Same

We left the history and evolution of MVS in the last chapter with the spectacular announcement in September 1992 of MVS/ESA SP 4.2 and other associated products. That was followed in February 1993 by the announcement of IBM's biggest "iron" ever—the 711-based ES/9000 9021s. In contrast with the trickle-down announcement style often employed in the past, IBM this time announced a comprehensive range of new technology processors, from the 711 uniprocessor up to the 8-way 982.

Unfortunately for IBM, "bigger, better, faster"—and more expensive—didn't seem to be what the marketplace was waiting for. That year, 1993, turned out to be the pivot point of IBM's fortunes. Revenue and earnings plunged, and IBM was finally moved to go outside its ranks for a new CEO—Lou Gerstner. After a bleak period of downsizing and reorganizations, IBM now appears to be on its way back. In this chapter we look at the MVS part of that turnaround.

Certainly not all of IBM's problems that peaked in 1993 were associated with MVS and its hardware platforms. However, it may

be instructive to consider the forces that were acting on that marketplace at that time. Understanding what happened to the mainframe marketplace in 1993 might lead to a better appreciation of more recent IBM announcements.

2.2. What Went Wrong

It's hard to come up with all the reasons for the decline of the mainframe. Maybe the most important was the distrust of accumulated centralized proprietary power. Perhaps administrators were falling behind in upgrading obsolete processors, thereby establishing fat targets for those advocating alternative platforms. However, there were less abstract economic pressures.

2.2.1. Forces acting on MVS

External forces that contributed to the perception of an MVS decline included:

- The rise of personal computers

  Early personal computers were often regarded as insignificant toys. However, the third generation of IBM-compatible PCs, built around the Intel 80386 chip, made it possible for PCs to manage memory in the tens of megabytes and in turn made Microsoft Windows a viable commercial operating environment.

  Continuing evolution has increased speeds and capacities, and driven prices down at a dizzying pace. At this writing, a 150 megahertz Pentium® processor sells for about $450. A recent newspaper had a typical advertisement for 16 megabytes of 60-nanosecond EDO DRAM for $99, and a 1.2 gigabyte hard drive for $139.[1]

  IBM failed to develop a coherent strategy for marketing commercially capable PCs and integrating them with what IBM planners always regarded as the mainstream product lines. Expensive IBM systems such as Series 1 and the 8100 were pushed aside as the PC became capable of doing the same tasks at a fraction of the price. The first 80386 machine did not come from IBM, but from the upstart

1 Bear in mind that the disk drive of the first System/360, the IBM 2311, had a capacity of just 7.5 *mega*bytes, and that $139 (in 1965 dollars) might not have paid the monthly maintenance bill on one drive.

Compaq. IBM's efforts went from failure to failure—the PCJr, PS/2 and MicroChannel, and early releases of OS/2.

When IBM finally produced a viable version of OS/2—Version 3 ("Warp"), Microsoft had already moved on to Windows 95 and Warp could not offer compatibility in either the user interface or the internal programmer's interfaces.

IBM's PC fortunes are not yet fully determined since the field is so fast-moving, but a visible strategy might help. Such a strategy, especially one delineating the relationship between MVS and PCs, seems to be evolving with the addition of enterprise server capabilities in OS/390.

- Local area networks

  Once PCs became accepted as "serious" machines, it followed that virtually all the data belonging to a department or a small establishment could be stored on a few high-capacity disk drives on a small number of PCs, connected by inexpensive telephone-grade wiring to other PCs for use throughout the business. The local area network (LAN) evolved rapidly from such a loose association to a structured and managed aggregate whose server systems ran under control of specialized network operating systems (NOSs).

  The communications capabilities of PCs and NOSs were initially incompatible with the mainframe's SNA network. Thus it took a long time and required considerable innovation for mainframes to become players in local area networking.

- Standardization of UNIX

  UNIX is an operating system of the same generation as MVS. However, it was developed originally by AT&T's Bell Telephone Laboratories as a streamlined control program for electronic telephone switching equipment, instead of as a commercial product for general-purpose computers. AT&T did encourage the latter use by making UNIX and its companion programming language, C, available to universities for teaching and experimentation purposes.

  With no acquisition cost and easy portability to any number of hardware platforms, UNIX became the everyday operating system and C the most common programming language in

many universities. However, given the combination of an operating system with available source code, a complementary programming language close to the hardware, and the nature of the university community, the development of variant versions of UNIX was inevitable. When it became apparent that those variants were mutually incompatible, standardization became an important goal along the path to commercial viability for UNIX. After an extended period of standards rivalry among different factions, a consensus emerged, standards were approved, and UNIX became perceived as the "open" system—as opposed to MVS as the archetypal "proprietary" system.

- RAID

  During the 1980s, the reliability of disk storage devices did not grow as rapidly as their capacity. An intrinsic failure rate of (for example) one hard error per year per gigabyte became unacceptable as storage capacities grew to the tens of gigabytes.

  A group of researchers at the University of California at Berkeley proposed an alternative: instead of challenging the hardware technology to deliver acceptably high intrinsic reliability, they would accept the inevitability of failures and use a combination of additional hardware along with software to deal with them. The combined technology they proposed was known generically as RAID, for Redundant Arrays of Inexpensive Disks.[2] At least six variations were proposed, each with different tradeoffs among speed, capacity, and error recovery. The most commonly implemented RAID implementation is known as RAID-5, which includes rotating redundant parity to allow reconstruction of data should any single disk drive fail.

  Once it became feasible to imbed the RAID logic in inexpensive disk controllers, a flood of RAID implementations made its way into the UNIX and PC marketplaces. It soon became commonplace for network servers to incorporate such RAID subsystems.

2 Once RAID became a commercial product, the meaning of the "I" in RAID became "Independent."

- The Internet

  The explosive growth of the Internet was fueled by the success of the World Wide Web (WWW). The "network of networks" had been around as long as MVS but had been important only in limited markets, until WWW provided the first link in organized access for casual users. Commercial products oriented to WWW now come out weekly, and everyone who wants one can now have a Web Page.

  Until late 1995, MVS and the Internet seemed to be strangers to each other.

- IBM's proprietary imperative

  The common theme of the foregoing items is openness of interfaces; the theme for IBM and notably MVS for most of its life was proprietary interfaces, closed to all but IBM and those who were favored as IBM's partners. Where the world had ASCII, MVS had EBCDIC. When the standard for local area network connections was Ethernet, IBM offered token-ring. When the near-universal standard for network communications was TCP/IP, IBM offered SNA.

  IBM's preference for proprietary interfaces made it easy for proponents of "open systems" to deride MVS as being out of step.

- The application development backlog

  Jokes abound on the subject of programs so old the source code had disappeared. As businesses grew and evolved, application users placed heavy demands on the application owners to add and change functions and to update the user interfaces of these applications. New concerns such as achieving readiness for the year 2000 increased the pressure on the application development resources, and there were always newly conceived application projects as well. The aggregate effect was a growing backlog of application development projects such that the deployment of a new application often takes two years or more after the requirements have been articulated.

  In fast-moving businesses, such an implementation delay is unacceptable. A common consequence is that those in need of an application may take matters into their own

hands—often by putting the essence of the application on a PC or UNIX system.

2.2.2. MVS in the bunker

Let's now summarize the problems of MVS as of 1993:

- A perception that MVS was closed and proprietary and therefore out of step and obsolete
- Vigorous competition from UNIX systems and rapidly evolving PCs, alone and in networks
- Non-participation in emerging trends
- Hardware that was expensive in many ways: acquisition cost, space, energy consumption, ...
- A diminishing lead in raw power
- Systems management complexity in areas such as installation and customization as well as disparate configuration, measurement, and tuning approaches across the system and its subsystems
- A slowing of IBM's pace of announcements, leading to a questioning of IBM's commitment to continuing success of large systems
- Limits to growth within a single MVS image
- Confusion about grandiose IBM initiatives such as SAA and SystemView

2.3. A Recipe for Turnaround

As IBM's mainframe fortunes hit bottom in 1993, IBM was well along in preparing its response. The turnaround was signaled by a blockbuster announcement on April 6, 1994—one day short of the 30th anniversary of the System/360 announcement. Before we go through the content of that announcement, let's review what the objectives of IBM's plan might have been.

2.3.1. Likely objectives for the 1994 announcement

- Solve the absolute and relative cost problems

  One of the attractive justifications for the movement to "dump the mainframe" was that just about any alternative solution, especially to a hardware configuration that was two or three generations obsolete, was less costly. A fraction of those who took such a course discovered enough virtue in an alternative platform to remain committed to their choice.[3]

  To preclude continued losses of this type, IBM needed to make the superficial cost comparisons more favorable before the alternative platforms became more competitive on their merits irrespective of price.

- Put Humpty Dumpty back together

  When OS/360 reached the marketplace in 1965, it was complete, including all access methods and the compilers necessary to build applications. (Subsystems such as CICS and IMS were never part of the operating system.) After the Consent Decree of 1969, IBM's June 23, 1969 "unbundling" announcement, along with IBM's organizational evolution, led to the MVS of 1993 existing in a multitude of separate pieces that were (in theory) merged only when a customer installed them.

  This problem was not solved in the 1994 announcement but was addressed in subsequent announcements.

- Bring space and energy requirements up to date

  Part of the forbidding atmosphere of the mainframe data center was the need for special environmental arrangements: space, distance limitations, raised floors, special power, special air conditioning, and, most special of all, chilled water and its unique problems and requirements.

  In addition to the cost of the infrastructure necessary to support a large mainframe, the cost of energy itself is a significant expense.

3 In many cases that cost advantage was illusory. The analysis, pushed by enthusiastic but (often) uninformed advocates, might overlook all costs but that of hardware and software acquisition and possibly energy. Comprehensive cost comparisons over an equipment life cycle are much closer and often turn out to favor the mainframe if the hardware is of a current generation.

- Focus system management on availability and service delivery of key applications and workloads

  As earlier editions of this book recounted, system management in MVS was organized around the measurement of system resource delivery to workloads. It took additional products to look at the workloads themselves. These products proliferated and evolved over time to become indispensable. However, the information revealed by these performance monitors had to be fed back manually to the operating system; correcting application and subsystem performance problems was complex and difficult.

- Simplify (a lot!) how work is defined and managed

  Managing an MVS-based system always seems to require a large number of overhead staff people. There are operators and their management, JCL analysis specialists to manage production batch workloads, help desk staff to assist on-line users, and a whole array of system programming and analysis specialties to be staffed—"real" system programmers who write code to extend and modify system function, performance and capacity specialists, specialists in the arcane language of SMP-E to manage system component and program product installation, ...

  If the management of the system could be made simpler—allowing a single operator to watch over multiple MVS images, for instance—the cost of running an MVS installation could be brought down. If the need for exhaustive SMP-E analysis prior to installing any new or upgraded programs could be reduced drastically, currency would improve. There are many such simplification opportunities, and all of them needed to be examined and evaluated if MVS were to continue as an economically competitive operating environment.

- Include SMS and application subsystems in overall system management approach

  Managing the optimal use of storage devices to accommodate an enterprise's data was viewed as a separate specialty. The introduction of DFSMS and its competitors made the job more efficient, but it still required and developed a separate set of skills. Similarly, a layer of specialists

grew around CICS, JES2 and JES3, IMS, DB2, VTAM, and their non-IBM counterparts. Each segment of this labor force became isolated in its specialty; workload balancing could not be attempted without extensive retraining investment.

Simplifying the job of subsystem management and fostering the development of portable skills was another key need for the future health of MVS.

- Implement a solid client-server development environment for developers, testers, and end-users

  Part of the reason for the application development backlog was the absence of an integrated environment for development of engineered client-server applications based on MVS as the server. All of MVS's strengths can be focused on file and data-base serving but business-critical applications demand more than function-based servers. They require closely coupled client-server applications in which the server "knows" the client and *vice versa*.

  The development environment of MVS until very recently supported only MVS-based terminals as user interfaces. PC-based graphical user interfaces (GUIs) did not integrate with applications on MVS. An integrating solution was needed.

2.3.2. The April 1994 announcement

The IBM announcements of April 6, 1994 provided the first round of comprehensive responses to the intrinsic and perceived problems of mainframes. In a huge, rich package, these were the highlights:

- New hardware technology—a midrange processor complex built on Complementary Metal-Oxide Semiconductor (CMOS) technology. At one-fourth to one-third the speed of high-end machines, CMOS could not at that time fully replace the faster but hotter-running and more complex bipolar technology where maximum single-engine speed was needed.

- A new architectural element—the Coupling Facility, an independent processor complex without channels but with

high-speed fiber-optic links which could be connected to multiple processor complexes supporting MVS images.

- A new multiprocessing architecture—parallel-coupled, in which application data is shared across up to (initially) 32 MVS images by means of ESCON channels and directors, and utilizing the Coupling Facility to ensure the integrity of the shared data. The multisystem configuration is called a *parallel sysplex*. Various software subsystems, notably CICS Version 4, were announced as "enabled" for parallel sysplex exploitation, and planned future enablement was announced for others.

- A new version of MVS—MVS/ESA SP Version 5, supporting the Coupling Facility and providing the operating system support for the parallel sysplex. A part of Version 5 is a new mode of operation for the System Resources Manager (SRM): goal mode, provided by a new Workload Manager (WLM) component. Chapter 7 describes WLM goal mode in detail.

- A new package solution—the Parallel Sysplex Offering, often known by the name of its core hardware as the Parallel Transaction Server (PTS). PTS (the IBM 9672 models E and P) is a single-box parallel sysplex, containing from one to eight CMOS processor complexes,[4] each a 2-way (dyadic) to 6-way tightly coupled single-sided multiprocessor. At least one of the CECs (or an LPAR within a CEC) needs to be configured as a coupling facility.

  The offering also included, at a comprehensive discounted price, all the software needed to establish an MVS parallel sysplex with CICS and the IMS Data Base Manager.

- A 10-way high-end system, the ES/9000 9X2.

- A separate CMOS Coupling Facility, the IBM 9674.

IBM added to the volume of the announcement and introduced some confusion by announcing the unrelated PowerParallel UNIX machines on the same day. This announcement package became known as IBM's "parallel announcement."

4 The CMOS processor complexes are known as CECs, for Central Electronic Complexes.

A great deal of instant analysis followed the announcement and the subsequent delivery of the PTS. Even though IBM speakers made numerous presentations pointing out the expected increases in CMOS speed, various pundits suggested that the new hardware was unsuitable for all but the most trivial of workloads. When the experiences of early users proved to be otherwise, the CMOS machines gained more respect.

2.3.3. Subsequent announcements

Following the April 1994 announcements, IBM continued to invest in the MVS marketplace, completing the response to the series of negative forces that existed in 1993. Some high points of those announcements:

- The 9672 (E/P) Parallel Transaction Server was re-announced as a separate machine, not just as part of the Parallel Sysplex Offering. There was wide recognition that the PTS was in fact a general-purpose machine even though IBM attempted to position it to favor its specific design target of high-volume simple transaction processing. It turned out that such a machine was well-suited to handle most batch and TSO workloads excluding those that were heavily CPU-intensive with single tasks.

- IBM renewed the name of its first DASD product, introducing RAMAC, a high-end RAID subsystem with a small footprint. A double-capacity RAMAC II was introduced in a later announcement.

- In September 1994, IBM announced the Parallel Enterprise Server, a new packaging of the 9672. The new models were designated as Model R. A single CEC was packaged in a very small cabinet. Initially, the R models used the same CECs as those in the PTS. Subsequent announcements in 1995 introduced the R2 and R3 models, with CPU speeds twice those of the original PTS, and larger CEC configurations, up to a 10-way. The 10-way RX3 has roughly the same processing power as the ES/3090 600J. Only the R models survive in the product line; multiple-CEC configurations may be built by bolting single-CEC models together.

- The evolution of MVS continued with the announcements of SP 5.2 and MVS/ESA SP 5.2.2. Key developments in these releases were the more complete integration of OpenEdition MVS, introduced in SP 4.3, along with enhancements to broaden the reach of MVS further into the enterprise. Additions for client-server application development, support of the Open Systems Adapter, UNIX XPG4 branding of OpenEdition, extensions to permit MVS to act as a LAN file server, all contributed to the repositioning of MVS as a broadly based open server. SP 5.2.2 also includes significant advances in dispatching and control of executable work units, reducing CPU contention and increasing the accountability of work units to the processes that created them. Some of these advances were necessitated by the need to provide full support of UNIX process structures in OpenEdition; regardless, they contribute to the flexibility of MVS's operating environment.

- The existence of MVS as a product (or at least a product name) came to an end with the March 1996 shipment of OS/390 Release 1. IBM completed its response to the list of MVS's deficiencies by re-integrating what had become a series of 30 or more disparate products into a base layer of function along with several optional layers. All of the components (now termed "elements") are integrated and tested together as a system. Formerly priced elements are now included in the base. Painstaking analysis of new release content can be bypassed with a simple replacement installation procedure.

  OS/390 prices are significantly less than those of the former MVS and associated products, and the release schedule is fixed at twice a year.

2.3.4. Another "September Spectacular"

Continuing the tradition of recent years, IBM put together another big announcement package on September 10, 1996. Highlights included:

- A new version of the Parallel Enterprise Server, called Generation 3, or G3. Since IBM used both R*x*2 and R*x*3 designations for the second generation, the 14 G3 models ranging from uniprocessors to 10-ways are known as R*x*4s.

Model designations are RA4 and R14 uniprocessors, RB4 and R24 dyadics, RC4 and R34 triadics, R44 through R94 (where the middle digit is the number of processors), and two ten-ways, the RX4 and the more powerful RY4.

The larger versions of these systems can replace ES/9000s as powerful as the Model 900, the largest of the 520-based systems, with the savings in space and environmentals that are characteristic of CMOS processors. They also can include an integrated cryptographic feature implemented through a CMOS coprocessor.

With this new CMOS generation, the speed advantage of bipolar implementations exists in only the latest systems. It seems to be a safe prediction that the CMOS speeds would surpass bipolar speeds within two or three more years.

- A different new version of the CMOS technology called Multiprise 2000 Servers. There are 13 models in this range. Seven of them are uniprocessors and there are two two-ways, two three-ways, a four-way and a five-way. The hardware configurations include processor storage of up to four gigabytes and a full range of optional features including the cryptographic coprocessor. The lower models of the range are more likely to be used as VM or VSE systems, but they can be configured with OS/390.

  The Multiprise 2000 systems are intended to be uncomplicated stand-alone configurations. They are not capable of connection to an external coupling facility and thus cannot be part of a parallel sysplex (except by using the test-mode ICMF code in a logical partition). Innovative packaging of mirrored SCSI-based hard drives with cache memory carved out of processor storage provide an "internal disk" facility with high performance and capacity of up to 288 gigabytes. A top-of-the-line model with five engines and 288 GB of DASD covers less than one square meter (about 10 square feet) of floor space.

  These systems are offered as an easy entry to the System/390 server world through a series of Enterprise Server Offering (ESO) packages. These combine pre-configured hardware and software (VM/ESA, VSE/ESA, or OS/390) along with IBM services.

- RAMAC 3 and the IBM announcement of four products made by Storage Technology Corporation (STK) but now marketed by IBM with some IBM-manufactured content. These are the RAMAC Scaleable Array Storage (STK Kodiak), a new model of the RAMAC Virtual Array Storage (STK Iceberg), and the RAMAC Electronic Array Storage (STK Arctic Fox).

  RAMAC 3 has approximately twice the capacity and up to twice the performance of RAMAC 2. The subsystem can connect to a 3990 Model 6 controller or can include a new 9390 controller which is the equivalent of one or two 3990s in about half the space of a single 3990.

- Completion of the X/Open XPG4 UNIX test suite and award of UNIX 95 Profile Brand Certification to OS/390 (as of Release 2) by X/Open Company, Ltd. This recognition, announced September 30, 1996, completes the journey started with the first release of OpenEdition MVS. Few believed at that time that it was IBM's goal for MVS to become in all senses a UNIX system, but now it is just that.

- OS/390 Releases 2 and 3. As promised in the initial OS/390 announcement, Release 2 was available at the end of September 1996 and Release 3 is to be available late in March 1997. Performance-related content in Release 2 includes Coupling Facility policy enhancements and GRS connectivity using the Coupling Facility with a "star" configuration as an alternative to the GRS ring configuration available to date.

  Release 3 continues to exploit the parallel sysplex and Workload Manager, with additional parts of the system enabled for load sharing and improved resource management. These include TSO, which now benefits from VTAM generic resources, allowing each LOGON to be directed to the system in the sysplex with the lowest workload. APPC/MVS work requests can be balanced across systems on a session basis, again by exploiting VTAM generic resources and WLM.

  The Workload Manager in Release 3 is enhanced to provide generic resource support for TSO and APPC/MVS, and to add DB2 stored SQL procedure support. Another new capability, "adaptive resource management," adds sys-

tem-wide and goal-oriented I/O priority management, address space management services providing dynamic management of application server address spaces, and management of individual work units within server address spaces based on business unit definitions. The latter capability eliminates the need to set goals for server address spaces.

A final change in Release 3 is to provide UNIX kernel services at all times. The UNIX subsystem is now started concurrently with the Base Control Program.

- SmartBatch for OS/390. This separately priced product is a successor to the BatchPipes offering. It extends the previous capabilities to include splitting batch jobs into parallel work units that can run across images of a parallel sysplex. It also includes a dynamic workload balancing capability for batch in a parallel sysplex.
- Sysplex Timer Model 2. This new model of the Sysplex Timer adds fiber-optic connectivity to increase the distance between processors and the timers. Through the use of cabling and connection features, a timer (or preferably a pair of timers) can be connected to up to 32 systems, the current limit for a parallel sysplex.
- Virtual Tape. This new subsystem combines tape cartridges and disk storage to project virtual devices and volumes. The full capacity of tape volumes can be exploited by combining logical volumes on a physical volume. The intermediate disk storage enables quick "remounting" of a data set that will eventually be stored permanently on tape, but which might be required in a job step just after the one in which it had been written.

2.4. Extrapolation

IBM's Chairman, Lou Gerstner, has been speaking often about "network-centric computing," in which high-bandwidth connections are exploited to deliver data, as complex as full-motion video, as required, to user work stations, with the necessary programs and subsystems residing on servers. (In today's environment the programs, as Java applets, would be downloaded along with the data.) The usual means of connection would be a public network connecting to an organized server facility such as the

World Wide Web. This type of connection could also exist in private networks.

This form of server-biased client-server computing is an ideal application for OS/390. MVS and its hardware platforms have the capability for high-speed multiple concurrent data transfer that will be the essence of such services. As MVS in OS/390 continues to adopt and integrate open communication standards, it will have no difficulty adapting to new connectivity options such as cable modems, direct-broadcast satellite, ADSL,[5] and ISDN (Integrated Services Digital Network).

In more traditional settings, the cost comparison between MVS and alternative platforms is much more nearly equal now, and likely to improve more as IBM's CMOS production increases with further improvements in price-performance. As OS/390 takes hold, the potential to reduce staff or to divert system programming staff to non-overhead functions can be realized.

Parallel sysplex will take off as well, once potential cost savings become well known; the potential efficiency of the architecture is realized; and maximum enablement of subsystems in OS/390 and widespread conversion from older versions of MVS to OS/390 is achieved. The use of Workload Manager goal mode will spread as soon as the instant mythology that grew up around it is debunked and CIOs start realizing that the management of workload service levels is indeed as simple as WLM promises.

With the obstacles to MVS growth and proliferation removed, IBM can turn its marketing efforts back to fostering the development of new OS/390 sites. A concerted effort to reach the universities has been needed for some time. MVS is an old operating system, but at the same time a thoroughly modern one, and very suitable for study in university curricula.

For a while it has been fashionable to denigrate the mainframe and to celebrate its departure from an establishment. (In most cases the displaced "mainframe" is a hopelessly obsolete 4341 or 3083 that today could be replaced by a P/390 PC-based MVS machine or an R/390 RISC/6000-based MVS machine.) This author believes that the fashion will change as the mainframe changes. Indeed the short-term results of the 1994 and subsequent announcements appear to be very posi-

5 Asymmetric Digital Subscriber Line, an advanced high-speed follow-on to ISDN.

tive. IBM's mainframe MIPS shipments grew by 40 percent in 1994 and were exceeded in 1995 by further growth of 60 percent—and IBM's stock price responded in similar fashion.

2.5. Summary

MVS has been a remarkably successful operating system. It has kept up with and responded to numerous generations of technology, architecture, and processors through more than a quarter century of evolution. Capabilities and attributes not dreamed of by its original designers have been integrated in its structure, and its reliability now approaches the ideal of continuous operation.

However, a once-substantial body of opinion holds that MVS is now obsolete, and that MVS systems can (and should) be replaced by some kind of assemblage of open systems in client-server configurations. What those advancing suggestions don't say is how all of the resource management functions of MVS can be accomplished in a network with no hub. The approach of more concentrated server environments, typified by the move to network-centric computing, will create greater need for bandwidth, concurrency, and resource management—all MVS strong points.

On the other hand, the requirements of today's environment are different and more demanding than those of a few years ago. IBM has devoted the resources to the appropriate and more or less timely enhancement of the MVS hardware and software environment. Continued successful competition for resources, which are limited even in IBM's laboratories, depends on the commitment that IBM's customers invest in the MVS environment. IBM seems to have concluded already that the demand for MVS remains strong, making a prodigious investment in OS/390. OS/390 has advanced beyond mere repackaging; Release 3 represents significant functional enhancement and simplification.

The [mis]perception that MVS is a difficult system to manage efficiently and effectively must be overcome if commitment to MVS is to be preserved and strengthened. Successful performance management is a necessary step in overcoming that perception and showing the value of MVS.

2.6. Chapter Questions

1. List five strengths and five weaknesses of MVS (pre-OS/390) that have become [more] visible in today's environment.
2. Concentrate on the weaknesses. State how each of these weaknesses tends to make installations abandon MVS.
3. How might each of these weaknesses be overcome by changes in MVS? If any of your weaknesses has no cure, set it aside as an *essential* weakness of MVS.
4. For the strengths you listed in question 1, how may each be made even more beneficial to the acceptance and long-term growth of MVS?
5. To what extent has your installation adopted the recent changes in MVS, including OS/390, Workload Manager goal mode, parallel sysplex, OpenEdition, and current levels of subsystems? If you have moved ahead, what has been the effect on the distribution of human resources in the data center? If you have not, why not?

Chapter

3

Physical Resources

MVS has no work of its own to do. Everything it does, it does on behalf of the workloads that run under its control. What, then, does it do? **MVS manages resources and the access of workloads to them.** Of course, an operating system does far more by providing services and handling exceptional conditions, including error recovery. In our concern for the performance management aspect of MVS, we'll ignore those functions for the most part. In this chapter we examine the physical (hardware) resources that MVS manages.

The end result of MVS's management of workloads is the interaction of those workloads with the physical resources of the system. We shall examine those resources and some of the difficulties workloads can encounter in trying to use them. In doing so, we will become aware of the need for the more abstract kinds of resources—virtual and logical—that MVS creates and supports in order to manage the physical resources.

3.1. CPU or Processor

Every program executes *some* instructions. The essence of a work unit such as a batch job, CICS transaction, or TSO command is the set of instructions to be executed. The work unit begins when the first instruction is executed, and ends (normally) when it executes a final instruction to return control to the operating system or the subsystem under which it runs.

MVS chooses a work unit to receive access to the CPU and dispatches it by executing a LOAD PROGRAM STATUS WORD (LPSW) instruction with the address of the first instruction of the work unit specified in the *program status word* (PSW) to be loaded. In the case of a subsystem like CICS, MVS dispatches the subsystem, and the subsystem's own internal dispatcher selects an internally known work unit to be given control of the CPU. Directly or indirectly, a productive work unit begins using a CPU by executing instructions.

To be useful, however, any program needs more than just the CPU. It must communicate in some way with the outside world, and ultimately with the human being who is using it. It usually needs data from some device in the input/output configuration or from some other work unit in the operating system environment. It may need some information known to the operating system, such as the time or date. Finally, it almost always requires working virtual storage, in addition to the storage needed for the program itself. (A well-written program in MVS does not alter its own storage. This discipline is the basic criterion for *reenterability*, an attribute identifying a program of which a single copy may be used by more than one work unit at a time.)

Each of these requirements is satisfied by some kind of interaction, outside of the current instruction stream, which is handled by MVS. A program usually requests an MVS service by issuing the SUPERVISOR CALL (SVC) or PROGRAM CALL (PC) instruction. Each of these instructions causes a change in the flow of control and causes an MVS service routine to be executed. When an SVC-invoked service routine completes its work, and upon the occurrence of certain other interruptions, the MVS dispatcher may get another opportunity to choose the next work unit to be dispatched (by performing the same operation as at initial dispatching, but with a restart address denoting the instruction following the SVC), and thus to receive more CPU service. (The dispatcher does not get to make the choice of who is to get the CPU next if an MVS service routine returns control of the CPU directly to its invoker, as is the case for PC-invoked services.) Many of the service requests (such as for an unbuffered I/O operation) cannot be completed immediately, so the requesting work unit is placed in a WAIT state pending completion of the request. The dispatcher is invoked again to ensure that the CPU does not go idle if any other work unit is ready to run.

I/O operations take hundreds to thousands of times as long as CPU instructions to complete, and the execution of I/O operations does not require the CPU except at initiation and completion. A substantial opportunity exists during an I/O WAIT period to dispatch a different unit of work and keep the CPU as busy as possible. (We have just defined multiprogramming. Of course, there is also the little matter of deciding which work unit gets to use the CPU at each such opportunity.)

The simple proposition of putting work into the CPU and letting it run to completion becomes complex in a modern multiprogramming operating system. MVS must have a way of knowing which work units are ready to receive service and which are waiting for the completion of some activity asynchronous with the CPU. The operating system must also determine, each time that the dispatcher has a chance to choose a work unit, which one is the most appropriate to dispatch next. This kind of decision is made hundreds or thousands of times per second. The lists of different kinds of work, and the rules for choosing the next dispatchable unit of work, are the objects and parameters of CPU performance management.

Before CPU management can be fully covered, an understanding of CPU configurations is necessary. We shall look now at the various types of CPU configurations supported by MVS. The treatment is in roughly historical order.

3.1.1. Uniprocessors

The simplest CPU configuration is the uniprocessor. As the name denotes, there is one PSW, one set of registers, one set of input/output connections, and one active instruction stream. Thus there is no simultaneity in processing, except what might go on beneath the "Principles of Operation" interface. Fast uniprocessors, beginning with the IBM System/360 Model 91 and continuing through most current large systems, have always had some level of overlap in their operation, such as between the instruction fetch operation and the execution of those instructions. What distinguishes a uniprocessor from a multiprocessor is the single set of architecturally defined resources, as opposed to those included in the engineering embodiment of the architecture.

MVS in a uniprocessor does not need to do those extra things we will see to be necessary in a multiprocessing environment. Thus

the throughput of a fast uniprocessor (assuming no constraint in other resources) can be higher than that of a multiprocessor built to the same architecture, using the same basic technology, and processing instructions at the same overall rate.

Uniprocessors under a given machine architecture can become faster in only two ways: (1) the basic technology (typified by machine cycle time) gets faster, and (2) the engineering designs get to be more efficient. An example of the latter might be to reduce the number of cycles needed to execute a commonly used instruction. The historical trend in uniprocessor speed improvement is about 15 to 20 percent per year. Unfortunately, the historical trend in demand for CPU cycles shows an increase of 30 to 50 percent per year. The gap is bridged by the use of multiprocessor configurations.

3.1.2. Multiprocessors

The word "multiprocessor" covers many different kinds of system configurations. IBM has employed the term "loosely coupled" multiprocessing to describe a JES3 configuration, in which several MVS CECs[1] are interconnected through channel-to-channel adapters. One might also extend the definition to encompass JES2 multiaccess SPOOL—CECs interconnected with shared DASD—and even more tenuously interconnected configurations.

Here, however, we shall use a more restrictive definition of multiprocessing—the kind originally named tightly coupled multiprocessing, or TCMP. In TCMP, at least two "processors" similar to uniprocessors share the same real storage and are controlled by a single copy of an operating system such as MVS. Another name for this configuration is *symmetric multiprocessing* or SMP, putting emphasis on the equal access by each processor to all of the non-processor functions of the complex. In the following discussion, we will see instances of asymmetric MP as well.

1 Since in this discussion, the term "system" might refer to an extended configuration and the term "processor" is part of the subject, we will sometimes use "CEC" (for Central Electronic Complex) or "processor complex" to denote the set of hardware associated with a single copy of the MVS control program.

System/360 Model 65MP

IBM's first production MP system of interest in the evolution to MVS was the MP65, a system using two System 360 Model 65s, modified to include some essential MP functions and "bolted together" via an "MP box" to operate under the control of a modified version of OS/360 MVT.

What are those MP functions? In a two-way MP system, each processor knows of only one other. If some significant event occurs that one processor does not know about, there is only one other processor that might be responsible. It may appear sufficient in two-way systems such as the MP65 to provide a simple "shoulder tap" instruction, so that each processor can cause an interrupt to the other. In practice, however, the inter-processor signaling mechanism is also a convenient alternative to normal operating system communication conventions, and the processor may issue a shoulder tap directed to itself. The SIGNAL PROCESSOR (SIGP) instruction introduced the notion of processor address, so that the interrupted CPU can learn the identity of the interrupter.

In every System/360-, System/370-, or System/390-based MP system, each processor must have its own copy of the byte address range 0–4095. This "low storage" area had unique architecturally designated functions in System/360: All old and new PSWs, the Channel Address Word (CAW), Channel Status Word (CSW), Interval Timer (obsolete since System/370), and Machine Check Logout Area are all assigned fixed addresses in low storage. Many of those functions continued in System/370, and later in Extended Architecture (XA) and in Enterprise Systems Architecture (ESA).

To maintain the independent operation of each processor, each needs its own image of the first 4K of low storage. This need is met through the addition of a new register, the *prefix register*, to each CPU, and of two new instructions. These two instructions were the first in the System/360 instruction set to break the solid bond between the addresses that programs use and the physically fixed order of an array of some storage (memory) device. SET PREFIX designates an address (divisible by 4096) in *absolute* storage[2] that will subsequently be known as real byte 0

2 An absolute address is the address the CPU and channels ultimately use for storage access. A real address is converted to absolute by means of prefixing, and a virtual address is converted to a real address by dynamic address translation.

for the current processor. The address translation covers real bytes 0–4095. The reciprocal translation is also done. Real storage references to the prefix page are translated to absolute page 0.

STORE PREFIX stores the address from the prefix register in the word of main storage denoted by its operand address.

System/370 Multiprocessors

In contrast with MVT, MVS was designed with multiprocessing support included from "day one." Access to unique but unsharable system resources was serialized through several software *locks.* (MP65 support had only one lock, thus spending proportionally more of its time than MVS "under lock," appearing at those times like a uniprocessor. The indifferent MP performance of the MP65, in the range of 1.6 to 1.7 times that of a UP, led to another pernicious legend—that MPs were intrinsically inefficient. MP efficiency has increased steadily since that time through engineering refinement, technology advances, and software improvements.) System/370 (and thus MVS/370) was still limited to two-way MP. Although prefixing, SIGP, and the MVS/370 locking structure could handle up to 16 CPUs, the System/370 engineering designs and the channel subsystem could not accommodate more than two.

Early System/370 MPs. The first System/370 MPs were the System/370 Models 158MP and 168MP. The 158MP employed cycle-stealing internal channels, while the 168MP had external channels. In each case, each set of channels was associated with a single CPU, eventually (in an early MVS enhancement) becoming known as a *channel set* and supported by a system availability feature known as *channel set switching.* Such a feature was necessary because many I/O control units (for instance, the IBM 3705 communications processor) could be attached to only one channel. If a CPU failed, its uniquely attached devices became unreachable by the operating system, and the availability benefit of the MP hardware was diminished. Channel set switching allowed devices attached to only one channel set and with unique device addresses to be reached from either CPU, keeping the system up and running instead of requiring an IPL when a CPU failed. Such devices were said to be *asymmetrically*[3] attached.

Considerations of channel set switching are typical of the painstaking configuration analysis and specification necessitated by

the architectural bonds among CPUs, channels, I/O control units, and devices in System/360 and System/370. In our consideration of the CPU resource, it is sufficient to note that MVS/370 was barely able to cope with two CPUs and their channels. XA, with its independent channel subsystem, was needed to simplify the connections and thus make MP beyond two-way feasible.

Succeeding the 158MP and 168MP were MP versions of the 3031 and 3033. In all of the System/370 MPs, an essential characteristic was *reconfigurability.* Each CPU, with its channels, could operate as an independent system under a separate control program (*physically partitioned* mode), or they could be combined as an MP under a single control program. The "MP box" that joined the CPUs performed an active function in *single image* (MP) mode, handling prefixing, timer synchronization, and inter-processor signaling. In *partitioned* mode, its function was limited to handling the apportionment of main storage.

The MP configuration provided added CPU capacity with a single control program, configuration flexibility, and availability. These advantages came at the cost of expensive MP features on both CPUs, additional switching features on I/O controllers, and the "MP box," which provided no functions of its own. System/370 multiprocessing provided no added I/O addressing capacity because of the limitations of symmetrical attachment, but when the number of channels and their aggregate data transfer rates were a system throughput limitation, MP systems, with a full complement of channels and sufficient control units, provided some relief from that constraint.

System/370 Attached Processors. In an effort to provide some of the benefits of an MP configuration at a lower price, IBM developed a type of processor complex called an *attached processor* (AP). In an AP (made available on the 158, 168, and 3033), there are two processors, but they are not partitionable. In most of the AP configurations, the secondary processor (the APU) has no channels, but the final version of the 3033 AP did have channels

3 Symmetrical attachment uses the two-channel switch feature of an I/O control unit or the string switching feature of some I/O devices to make the control unit address (and thus the addresses of the attached devices) the same on each channel set of an MP. With more complex switching options, alternative (symmetrical) paths could be provided on each channel set. Asymmetric connection results when these features are not used or are unavailable.

associated with the APU. (We shall consider the AP in its original concept, without channels.)

The AP configuration had virtually no availability advantage but did provide added processor power for workloads that were CPU-intensive with less-than-typical I/O needs. When APs were installed for general-purpose use, it soon became clear that I/O was a significant bottleneck. The APU could not issue the START I/O FAST (SIOF) instruction. The I/O supervisor (IOS) on the APU had to SIGP to the CPU to initiate the I/O. System/370 architecture required that the (main) CPU take all of the I/O interrupts. In today's terms, the AP was inherently incapable of being a "balanced system."

Dyadic and Triadic Processors, and Beyond ...

Recall that the 3081 was initially announced as a System/370. In that context, its configuration was yet another multiprocessor variation. The new term was *dyadic*,[4] denoting two processors in a single box with two associated channel sets. The System/370 appearance of the 3081 was similar to that of a 3033 AP with channels on the APU: an "MP" that was not separable into two uniprocessors, but with a full MP complement of channels organized as two channel sets, permitting symmetrical configurations to be moved intact from 3033 MPs.

When the (true) Extended Architecture nature of the 3081 was made known, the overly complex System/370 structure could be swept aside. There were simply two CPUs, each equally capable of using the independent channel subsystem. The association of channels with CPUs was discarded, and there was no reason why a third or fourth CPU could not be added to the dyadic configuration. Such an addition was made in the generation following that of the 3081 in the 3090-300E. Conceptually, the triadic is no different from a dyadic. These are all examples of one-sided multiprocessors, configurations that can't be physically partitioned.

One-sided MP configurations continued to evolve in the ES/9000 and the 9672. A maximum ES/9000 711-series side is now a 5-way (in the 9X2), and the 9672 RX3 and RX4 are 10-way CECs.

4 Webster's New Collegiate Dictionary defines "dyad" as "**1:** PAIR specif: two individuals (as husband and wife) maintaining a sociologically significant relationship **2:** a meiotic chromosome after separation of the two homologous members of a tetrad...."

Dual Processors

In the intermediate range of processors smaller in capacity than the 3081, another multi-CPU variation appeared. The largest models of the IBM 4381 were *dual* processors. They were similar in concept to dyadics, but the channels were associated with CPUs, as are System/370 channel sets, and were cycle-stealing ("integrated") channels. The XA requirement of independence (of channels from CPUs) was met with additional microcode, suggesting potential inefficiency of the 4381 design in XA mode. The ESA models of the 4381 continued the "integrated channels" approach. The air-cooled frame (9121) models of ES/9000 went well beyond the 4381 in supporting the System/390 architecture with separate channel processors.[5] Indeed, the 9121 design resembles that of the IBM 3090 more than it does that of the 4381.

3084 and 3090 Multiprocessors

Prior to the announcement of the 3081, the biggest MVS systems were always "full" MPs. They had two CPUs and two channel sets, and were partitionable into two independent quasi-uniprocessors. With the 3081, such an arrangement was not possible, since the dyadic already had two CPUs and two channel sets. After XA was announced, it became possible to announce the 3084. A 3081 was one side of a 3084; the other side was originally a spectacularly expensive "upgrade kit," but eventually two 3081KXs could be fused into a 3084QX. (Perhaps as important, they could be unfused as well. Many installations made successful and non-disruptive migrations from MVS/370 to MVS/XA by exploiting the ability to partition the 3084 into two 3081-equivalents.)

The designs of successor systems built on the two-sided model of the 3084. The 3090-280E, S, and J and the 3090-250S and J were basic MPs like the 3033MP, composed of two partitionable "sides," each equivalent to a 3090-180E, S, or J or 3090-150S or J, respectively. The 3090-400 and later the 400E, 500E, and 600E (and their S-series and J-series counterparts) were composites of dyadics and triadics, as indicated by their model numbers. The 3090-380S or J was a composite of a uniprocessor and a dyadic.

5 The May 1991 issue of the *IBM Journal of Research and Development* (Volume 35, Number 3) covers the 9121 design exhaustively in several articles.

Each two-sided system retains the physical partitionability of the 3084, but that becomes a less important attribute with PR/SM . Similar in purpose to Amdahl Corporation's Multiple Domain Facility (MDF) and Hitachi Data Systems' Multiple Logical Processor Facility (MLPF), PR/SM allows *logical partitioning* of a processor complex into as many as ten independent system "images" per side. Thus a hardware configuration can support multiple operating systems running concurrently without the need for physical partitioning or a software *hypervisor* such as VM/ESA.

ES/9000 Configurations

There are three major model series in the ES/9000. The 9122 models are relatively small, rack-mounted configurations, either uniprocessors or dyadics. The mid-range models are the 9121s. The most recent models are denoted as "511-based," consisting of 10 models ranging from uniprocessors to 4-way MPs.

Topping the ES/9000s are the 711-based 9021s. These comprise the third generation of 9021s, supplanting the 340-based and 520-based earlier models. In this series there are 12 models ranging from the 711 uniprocessor to the 10-way MP 9X2 model. (There are choices within the range between one and two sides with the same number of processors.)

The IBM 9672

The original E- and P-models of the 9672 (the Parallel Transaction Server, PTS) were part of the Parallel Sysplex Offering but they do not need to be run as a parallel sysplex. In the Parallel Enterprise Server (PES) R-models, each is a single processor complex or CEC, ranging from a uniprocessor to a 10-way. A PTS contains from one to eight CECs, each originally a 2-way to a 6-way. PTS configurations can also be based on the R-model CECs, thus accommodating as many as 80 CPUs in a cabinet the size of a basic 4341.

3.1.3. Sysplex and Parallel Sysplex

In MVS Version 4, IBM began promoting the idea of a sysplex—a loose confederation of systems united through a single console message stream as well as through the multisystem capabilities of the job entry subsystems. In the same time frame, multisystem configurations of IMS and CICS grew in popular-

ity, with the CICS implementation being denoted a "CICSplex." These implementations moved in the direction of a single system image across multiple MVS systems, but didn't quite get there.

With MVS Version 5, IBM began supporting another distinct multiprocessing architecture. This architecture, the *parallel sysplex*, went beyond the basic sysplex by adding the coupling facility and all that it made possible. A parallel sysplex is more loosely coupled than traditional tightly coupled MPs, but considerably more tightly coupled than what exists in JES complexes or CICSplexes. Since specific hardware is required to implement a parallel sysplex, it is rightly viewed as a new generation of system architecture.

In contrast with other MP architectures, a parallel sysplex contains multiple copies or images of the operating system, up to 32 on separate hardware configurations as of OS/390 Release 1. However, there are several unifying programs—including the global workload manager, GRS, XCF, console support, and the coupling facility code, that provide a comprehensive implementation of a single image.

Why was Parallel Sysplex Needed?

Very simply put, those responsible for an installation that grows beyond the bounds of a single multiprocessing system incur a significant burden of complexity in considering how to accommodate that growth. The choices are not easy. Suppose the backlogged and anticipated growth is 50 percent of the current capacity. Here are some possible choices:

- Acquire another large system, about 75 percent as powerful as the current one. If possible, reconfigure both to be of approximately equal capacity. Operate them in a basic sysplex configuration to secure some limited single-image benefits. The extra capacity is needed to accommodate the overhead of two-system operation and to deal with possible additional latent demand that cannot easily be estimated in a highly constrained system.
- Determine what work can be moved to non-MVS platforms and acquire some number of small systems to take on that work. It's likely the growth crisis will reappear in a couple of years.

- Dump the mainframe and distribute everything to small systems. The cost will be staggering.

The problem of growth may be characterized in two areas: scalability and granularity. If an architecture is scaleable, the addition of each increment of capacity has (nearly) equal effect. Granularity means that the increment of growth is conveniently small each time that growth is needed.

Conventional MP architectures are not scaleable beyond the maximum the vendor implements: an ES/9000 9X2 would not benefit enough from adding an eleventh processor (if it could be done) to justify the cost. The scalability has improved through the process of invention and design refinement from a level of two up to 10 in the IBM implementations, and to 12 in the case of Hitachi Data Systems. Even if a breakthrough in MP efficiency were to be realized and the architectural limit of 16 were to become feasible, the limit is eventually reached and one can go no further.

Although scalability is a characteristic problem at the high end, granularity is more of a problem at the low end. Smaller installations cannot afford to procure excess capacity beyond current needs, so a small increment of capacity is very desirable.

Parallel sysplex addresses both these concerns. Because the degree of coupling is much less than in multiprocessors, a parallel sysplex approaches linear scalability. Adding the twentieth system to the sysplex adds almost as much capacity as adding the second. As for granularity, if a small increment of growth is needed, a single 9672 3-way can be upgraded to a 4-way. If a large increment is needed, one can add three or four 9672-RX4s—or even an ES/9000 9021-9X2.

In addition to the 9672, ES/9000s based on the 511 or 711 uniprocessors can be equipped with the necessary coupling links and the microcode to support the coupling facility and therefore can be part of a parallel sysplex.

Elements of a Parallel Sysplex

As shown in Figure 3-1, parallel sysplex is more than just a number of MVS images tied together through a coupling facility. It also is a means of sharing data reliably across multiple MVS images. The means of that sharing depends on several elements:

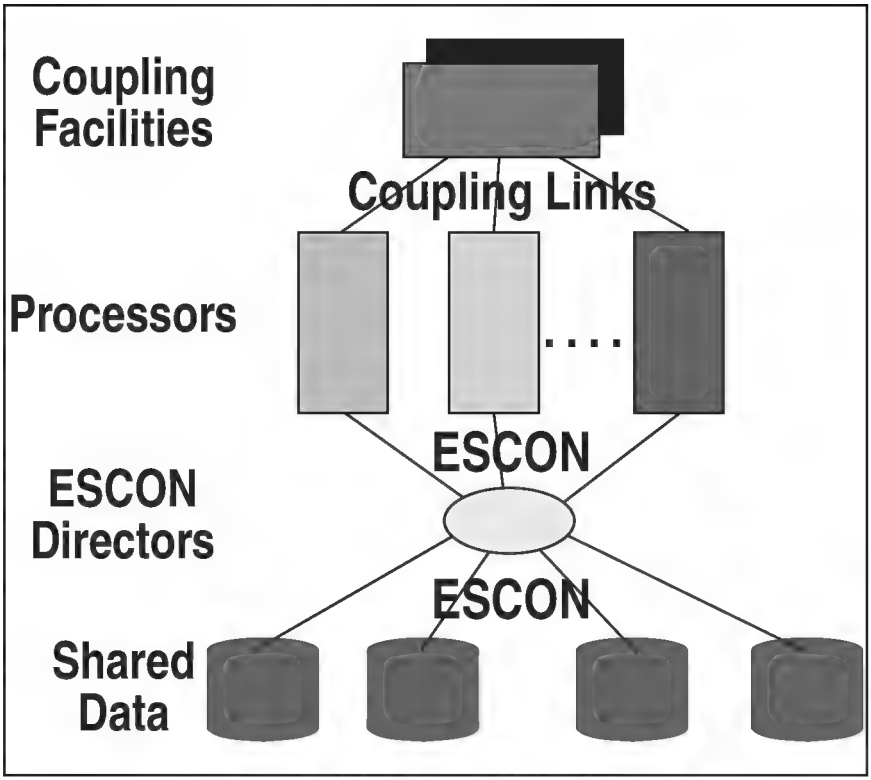


Figure 3-1. The Parallel Sysplex

- ESCON channels and ESCON directors to provide the data paths between each image and all the devices to be shared.
- One or more (to preclude a single point of failure) coupling facilities and the high-speed fiber coupling links connecting them to each MVS image
- Access methods or subsystems that are aware of parallel sysplex data sharing and the coupling facility, and use the coupling facilities at a minimum to control the integrity of the shared data. They may also further exploit the coupling facilities to cache other control data or to use a coupling facility as a shared I/O cache.

The parallel sysplex appears to remove growth constraints and increase the amount of hardware that a given staffing level can manage effectively. Once those responsible for a parallel sysplex

understand the value of making all the systems look the same ("cloning") additional benefits can be realized:

- simplification of system and product installation and maintenance
- immediate benefits of system-managed load balancing
- continuous operation with the ability to perform maintenance on any image without shutting down the others

3.1.4. CPU—Summary

In every MVS system image, regardless of the CPU configuration, there is only one dispatching queue. Whenever an MVS routine that exits *via* the dispatcher completes execution, the dispatcher is invoked on that CPU. In Chapter 4 we'll examine dispatching priority and its control, as well as the elements that make up the dispatching queue.

3.2. Real Storage

In many of the older CPU configurations, it was nearly impossible in practice to use all of the CPU power productively. The bottleneck most likely to have inhibited full CPU productivity was that of insufficient real storage.

Real storage in MVS is organized into page frames of 4096 bytes each. Programs in MVS refer not to real storage, but to virtual storage addresses. (An exception occurs in the input/output subsystem: I/O data addresses are absolute, and *access method* routines translate virtual addresses to absolute addresses as part of channel program construction.) Hardware translates the virtual references to real addresses through dynamic address translation. MVS manages real storage on the basis of demand; the Real Storage Manager assigns frames from an available frame queue (AFQ) when they are needed. Frames are reassigned (stolen) when the AFQ length falls below a threshold, or when a need for frames cannot be met from the current AFQ.

Associated with each page frame are three indicators used by MVS. The *reference bit* is set by hardware whenever an instruction "touches" any byte in the page, whether as code or data. The *change bit* is similarly set by hardware when instruction execution causes so much as a bit in the page to change. An MVS routine examines the reference bit periodically, resetting it when-

ever it is found to be on. The information conveyed by the reference bit is used to set the value of the third indicator, a 1-byte counter called the *Unreferenced Interval Count* (UIC). When the UIC update routine turns off the reference bit, the UIC of the page is set to 0. If the reference bit is still off from a previous update, the count is increased by the number of real seconds since the last update, to the maximum of 255.

UIC is used as a criterion to select which frames are in use and which are not, so that idle frames may be reassigned to fill more urgent needs. The change bit is used to determine if a currently valid backup copy of the page exists on the DASD that forms the paging subsystem (*auxiliary storage*). If so, the frame can be "stolen" with impunity. Otherwise, a backup copy must be sent to either expanded storage or auxiliary storage before the frame may be reassigned.

To understand the great importance of carefully planned real storage management in MVS, consider some of its uses:

- Residence for pages of the system with virtual addresses the same as their real addresses ("V=R").
- Residence for pages of programs that need to run in V=R mode. V=R programs and system routines were prevalent in the early days of SVS and MVS. As fear of the unknown diminished, dynamic address translation was recognized as innocuous (if not beneficial), and V=R strictures were removed as code was rewritten from the MVT base.
- Residence for fixed pages. Pages are fixed (removed unconditionally from eligibility for reassignment) because addresses within them must be invariant across I/O operations that work with real addresses and usually take a long time to complete. The short-term page fixing associated with I/O need not be expensive in real storage. Only buffers associated with an I/O currently in progress need be fixed.

  Pages are also fixed for other reasons. To enhance performance in one part of the system, usually at the expense of performance elsewhere, pages may be long-term fixed. Unless done with great care and with full analysis of the process whose pages are to be fixed, including its other resource needs, such page fixing often produces little benefit at considerable cost. The use of storage isolation or the

adaptive management performed by the Working Set Manager or Workload Manager are more appropriate techniques for this purpose.

- Residence for system data. The control blocks and data queues through which MVS manages resources are in this category. Some are fixed, but many simply remain resident because of their high activity.

- Residence for the working sets of active address spaces. "Working set" is a difficult term to define, because it varies depending on the time span of interest. In any time interval of interest, it is the amount of real storage (usually denoted in pages) a program needs to complete the current activity without adding another page frame. At the lowest (instruction) level, this amount is often a single frame, but one might imagine a worst case in which eight frames are needed. A single TSO transaction may need 50–200 frames.

  "Working set" is a concept that maps well to the behavior of a single program. Most programs have the property called *locality of reference*—instructions executed later are not far in virtual storage from those executed earlier, and the data they access lie within a small span of virtual addresses. Consequently, they make reference to only a few real page frames as well.

  As subsystems such as CICS, serving many online users, became the norm in MVS, the working set concept became less useful. Growing use of client-server designs built on APPC/MVS, Open Edition, or other frameworks will reinforce this trend in the case of server address spaces. Successive requests are likely to be unrelated, so there is little continuity of virtual or real storage reference from instant to instant.

  Consequently, we cannot speak of a "working set" for a CICS subsystem in the strict sense. Rather, we regard the working set for such a subsystem as the number of pages it needs to have bound to real storage frames to avoid sustaining a damaging page fault rate. This is not a crisp definition, but rather one element in a series of tradeoffs between the (resource or dollar) cost of providing service and the effect of a particular level of service on the larger busi-

ness. In a later chapter we'll discuss *storage isolation*, an MVS service that allows such storage binding to be selected and controlled, and Working Set Management, an SRM function that achieves most of the benefits of storage isolation without the use of external controls.

- Residence for the working sets of inactive address spaces. As MVS evolved, there were several time periods in which the focus of interest swung between saving real storage and exploiting it, between using the paging subsystem heavily and avoiding its use, and between throughput and response time. MVS today retains traces of the extremes of those swings—in default parameters that oppose each other, and in basic mechanisms that are equally in conflict. It appears that real storage constraint (at least in today's ES/9000s and parallel servers) is not a current problem, so that the mechanisms encouraging real storage residency for inactive address spaces, as well as for inactive pages of active address spaces and system data areas, are now dominant.

  This use of real storage began in MVS/SE1 with the introduction of logical swapping. In that time frame, 168MPs with 16 megabytes of real storage started to become available. The full real storage complement was often more than the system could use, and the chronic problem of TSO's paging conflicting with the response time needs of other workloads was solved by exploiting the surplus real storage to maintain inactive TSO users in real storage.

3.2.1. MVS/370 and the "16-megabyte line"

In System/370, 16 megabytes was the architectural upper limit of both virtual and real storage addressing. As the 3033 replaced the 168 at the high end of the line, and as MVS's internal constraints were gradually eliminated, systems started to become unbalanced again, this time with real storage in short supply. The problem was particularly acute in 3033MPs. Two 16-megabyte 3033 uniprocessors (UPs) could be made into a symmetrical MP limited to only 16 megabytes, leading to a drastic real storage shortage. The difficulty was temporarily overcome with the "extended addressing" feature on the 3033 which used a previously reserved bit in a dynamic address translation table entry to double the size of real storage to a maximum of 32

megabytes. (Use of another such bit stretched the limit to 64 megabytes in later CPUs.)

Easing the real storage constraint led to a new class of problem in MVS/370: Which pages could go to page frames in the extended addressing range? In System/370, the channel command word (CCW) format was limited to a 24-bit real address, so normal I/O access methods could not exploit real storage "above the line." (The operating system made use of the Indirect Data Address Word [IDAW] to address upper real storage, so paging I/O could exploit extended addressing.) Consequently, MVS/370 beginning with SP1.1 had to move pages from frames above 16 megabytes to frames below the line when those frames contained buffers for active I/O. This added system overhead, prominently reported in monitoring programs such as RMF, became significant as systems were loaded to full capacity with TSO users and interactive subsystems. As with many other System/370 growth limits, the partial benefit of added real storage became more complete only with Extended Architecture.

3.2.2. Real storage in MVS/XA and MVS/ESA

In System/370 Extended Architecture and Enterprise Systems Architecture (and therefore in MVS/XA and MVS/ESA), the real storage limit was extended to 2 gigabytes. The XA compatibility goals required that System/370 channel programs operate unchanged in XA mode. However, System/360 and System/370 channel command words are limited to real data addresses of 24 bits. XA and ESA thus have two CCW formats; the original 24-bit CCW is denoted format 0, and the newer 31-bit version is called format 1.

Several MVS/XA and MVS/ESA access methods, as implemented in the Data Facility Product (DFP) portion of the operating system, have already been re-implemented to create format 1 channel programs, and more have followed. Therefore, the corresponding I/O operations are not restricted in the real storage areas used for input or output. Antique MVT or MVS/370 programs that build their own CCWs still need to place those channel programs and buffers below 16 megabytes (real), or the operating system must move pages above and below "the line" as MVS/370 had to do.

3.2.3. Central storage and expanded storage

As we consider the uses of real storage, it becomes clear that much of it is occupied by relatively inactive pages. The alternative, to place those pages on auxiliary storage, leads to significant delay from the paging subsystem when those pages are needed later. However, the demand for such standby storage increases greatly as CPUs grow more powerful and become capable of supporting much larger workloads.

When System/360 was first announced, and even when System/370 was first announced, the storage limit of 16 megabytes seemed more than generous. However, with the explosion of growth facilitated by MVS, soon it became clear that 16 megabytes was not enough. When XA was announced, the 2-gigabyte real storage limit was welcomed, but with fewer predictions of its being excessive.

To provide a means of accommodating inactive pages without the reactivation delay of auxiliary storage or the cost of "real" storage, a new solution was devised for the 3090 line and eventually added to the CPU architecture in Enterprise Systems Architecture (ESA).

The solution is called *expanded storage* (ES). Once again, the introduction of a new element added complexity and caused nomenclature to be changed. IBM stopped using the term "real storage" and has substituted *processor storage* for it. In systems with expanded storage, processor storage is now divided into *central storage* and expanded storage. Central storage is, of course, what we used to call real storage.

Expanded storage is not accessible to the I/O subsystem[6] or to ordinary instructions in the CPU and is addressable only as pages, not as bytes. Expanded storage is large and fast. Sizes of up to 8 gigabytes are already available. In the IBM ES/9000-9021s, where expanded storage is implemented differently from central storage, it performs at about half the speed of central storage. In the IBM ES/9000 Model 9121s and the IBM 9672 Parallel Servers, a single increment of up to 8 GB of processor storage is divided between central storage (with a maxi-

6 The 520- and 711-based ES/9000s have a hardware unit called the Interconnect Control Element, or ICE, that manages both expanded storage and the channel subsystem. It seems reasonable to speculate that at some future time, the proximity of these two subsystems in the ICE might lead to a direct functional connection between them.

mum of 2 GB per LPAR) and expanded storage. The same approach is used in some non-IBM processors. In systems with partitionable storage of less than 2 GB, the use of expanded storage should be justified on the basis of functional need. However, some of the real storage management functions of MVS/ESA tend to be more efficient when expanded storage is available. A close choice should be resolved in favor of more expanded storage when there is a significant swappable workload. For systems with more than 2 GB of storage, the use of expanded storage or logical partitioning are the only ways to get the full benefit of all the installed storage.

The only expanded storage operations supported are "page in" and "page out," instructions to transfer pages synchronously between expanded and central storage. (A MOVE PAGE instruction with a wider choice of operands and with considerably better performance is supported on 3090J models, ES/9000s, IBM's CMOS machines, and current machines from other vendors. It is a prerequisite for use of some MVS/ESA facilities.)

In MVS/XA and MVS/ESA, expanded storage is managed by the Real Storage Manager (RSM) with guidance from the System Resources Manager (SRM). In contrast, VM/SP manages expanded storage as a fast paging device, and VM/ESA simply apportions expanded storage to virtual machines to use as the operating systems in those machines direct. MVS/ESA adds *hiperspace* services providing additional direct and explicit access to expanded storage by application programs and subsystems.

Expanded storage meets the need for standby storage of inactive address spaces or of individual pages without devoting full-function central storage to that purpose. Consequently, the central storage resource can grow at a slower rate than previously, and the 2-gigabyte limit is therefore safe for a few more years. The architectural limit of expanded storage is 2^{32} pages, or 1.76×10^{13} bytes (16 terabytes). It is with and because of expanded storage that chronic real storage shortage in XA and ESA is being called "a thing of the past." Of course, not all MVS systems have expanded storage.

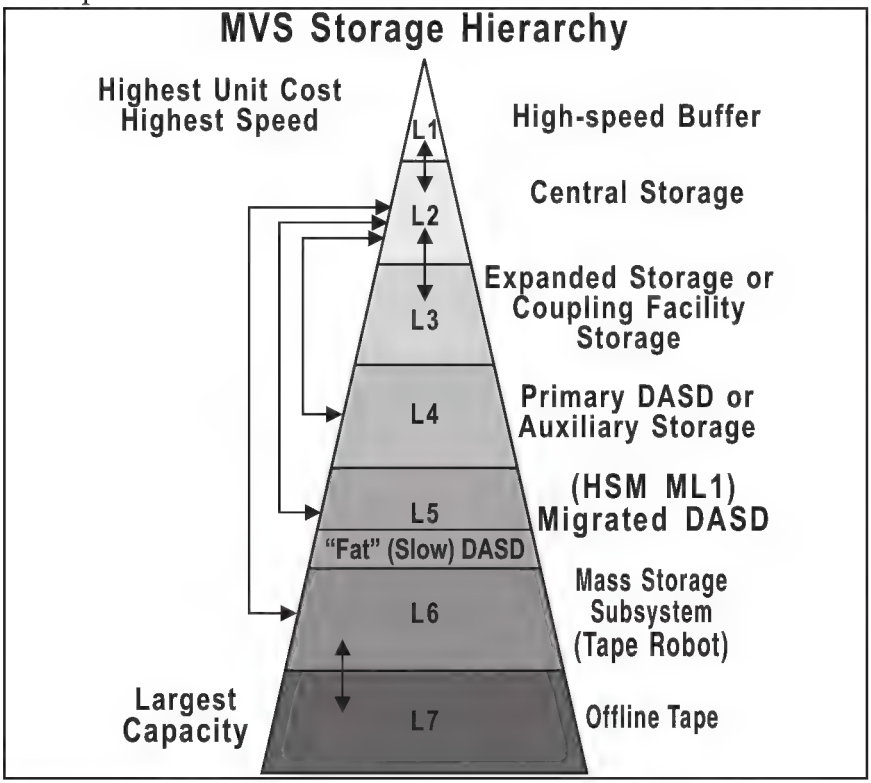
MVS/ESA's options for replacing I/O operations with the use of processor storage may well bring back real storage constraint unless careful planning (including having a generous amount of processor storage for contingencies) is done. The

widespread use of "multiple image" options, such as IBM's PR/SM, HDS's MLPF, and Amdahl's MDF, will further engender processor storage constraint.

We will therefore consider the problems of real storage constraint in detail throughout this book.

3.2.4. The MVS/ESA storage hierarchy

It's sometimes useful to picture the various levels of real storage in an MVS/ESA system as forming a pyramidal hierarchy,[7] with the fastest and most expensive form of storage at the top and the least expensive and slowest form at the bottom. There is much to learn from considering the ways in which data moves between levels in the hierarchy. Figure 3-2 shows the levels and the movement paths.



7 The notation used is consistent with that of commonly seen representations of storage hierarchies, but the names of levels and examples are arbitrary.

Central storage (L2) is addressable to the level of a single byte by instructions and is associated with the CPU and its high-speed buffer (HSB).[8] Central storage is also the source and destination of I/O to and from the channel subsystem.

Expanded storage is not addressable from ordinary instructions and is managed by the operating system in full pages (4096 bytes) only. Expanded storage may be slower than central storage (but much faster than any I/O device) and cheaper than central storage (but more expensive than a channel-attached storage device). It therefore has a natural position as the third level (L3) of a storage hierarchy, following the processor high-speed buffer (L1) and central storage (L2), but preceding channel-attached devices (L4 and below). At a parallel position is the structure storage in a coupling facility. Depending on the implementation, coupling facility storage may be slightly slower than expanded storage but still within the same order of magnitude.

Data movement between L1 and L2 is managed transparently by hardware. All other movement between levels of the hierarchy is done by the operating system. Movement between L2 and L3 (expanded storage) is called *page movement*, and takes the place of paging and swapping using auxiliary storage (L4) in systems lacking expanded storage. Movement to and from L3 coupling facility storage is initiated by specialized instructions in the System/390 instruction set as augmented for parallel sysplex.

Movement from L3 (expanded storage) to L4 is called *migration* and requires L2 storage as a temporary holding location because I/O cannot take place except to and from central storage. There is no movement from L4 to L3. Movement between other lower levels also requires I/O and thus uses L2 as an intermediary. An exception exists when the physical medium can be transported (as between L6 and L7).

The prior discussion suggests that L2 (central storage) is the busiest level of the storage hierarchy. This is so, and the assurance of an adequate supply of central storage is the essence of real storage management. In a well-managed MVS system with a mixed workload, there are two mechanisms working to ensure availability of central storage as it is needed:

8 The high-speed buffer in today's systems may comprise two or even three levels. In the 9672 R models, there is an HSB on the processor chip and a second-level buffer on an adjacent chip, much like the structure found in most PCs with Pentium processors.

- *Swapping* moves a whole inactive or less-preferred address space from central storage to expanded storage, or to auxiliary storage if expanded storage is not available. What was swapped out will eventually be swapped in when the address space is ready to run and MVS's controls allow it back in. About 50 to 500 frames of central storage are freed by a typical swap-out.

- *Page stealing* moves a few pages at a time from central storage to expanded storage, or to auxiliary storage if expanded storage is not available. In earlier versions of MVS, what was paged out would eventually be paged in, one page at a time, as reference was made to pages not currently in central storage. About 10 to 50 frames of central storage were freed by a cycle of page stealing, with the pages normally coming from several address spaces.

  Since MVS/ESA SP 4.2, page stealing and the resolution of page faults have become more efficient with blocking of both page-outs and page-ins.

It can be seen that swapping is more efficient than page stealing as a means of replenishing the AFQ, if the swappable address spaces are available. Systems with only nonswappable work (such as dedicated production CICS systems) cannot enjoy the benefits of swapping and therefore require somewhat higher levels of central storage than systems running mostly batch and TSO or a mixed workload.

3.3. Input/Output Resources

We turn now from real storage as a source of performance problems to the second most likely source in MVS/370, and probably the main tunable problem in XA and ESA. That source is inefficiency, contention, or a simple lack of resource in the I/O subsystem. MVS/ESA has several mechanisms designed to lessen vulnerability to I/O contention and delay, but a substantial number of MVS systems today do experience I/O problems.

Several kinds of I/O-related delay affect MVS workloads:

- Direct delay in waiting for I/O completion. Such "active I/O" delay is often a sign of inefficient I/O: too few buffers, extended use of the DASD SEARCH command, fragmenta-

tion of data on devices, or failure to use more efficient alternatives to conventional I/O.

- Delay waiting for access to devices. "Queued I/O" delay is most often caused by poorly planned sharing of direct access devices, either between systems, between workloads, or within a single workload.

  If one of the workloads sharing a device is much more important to the installation's business than the others, its data sets may need to be moved to a device less susceptible to contention. Such "I/O tuning" should take place in conjunction with responding to a service discrepancy for the workload, rather than being based on a search for "problem" devices. Often, devices that look "troublesome" on the basis of utilization or queue length are serving workloads designed for just such efficient I/O, or workloads with less stringent service needs, and are not causing unacceptable levels of delay to more important workloads.

- CPU delay caused by the I/O. CPU cycles are used to set up each I/O and to deal with the hardware interrupt that marks its end. In certain processors, such as the IBM 4381 family, the channel subsystem is not wholly separate from the CPU. In such "integrated" or "cycle-stealing" channel configurations, CPU cycles are not available while the channel is accessing storage. Cycle-stealing channels are not found in today's systems.

  Regardless of whether CPU cycles are stolen to use for the I/O operation itself, "front-end" and "back-end" activities occur at the highest dispatching priority in MVS (global SRBs) and thereby interfere with CPU access for all workloads. A general goal of minimizing the number of avoidable I/O operations is one of the few MVS performance management absolutes.

We shall divide I/O into several classes and concentrate on only a few of them:

- *Source/sink devices.* These are devices that bring simple input to the system, take output from the system, or both. Printers and card readers and punches are examples. These used to be called *unit record* devices, the unit being a card or a line of print. Today's printers scarcely fit in this mold, be-

ing high-speed, all-points-addressable page printers; very few systems now have card I/O. Locally attached terminals fall into this category as well, but we choose to regard them as communications devices because of the human user.

Source/sink devices are usually managed by the Job Entry Subsystem (JES) and are rarely a source of system performance problems.

- *Console devices.* These are (usually) locally attached devices for use by the operations staff, and are limited to terminals and printers. Variations today include pseudo-devices managed by automated operations packages, personal computers emulating consoles, and systems with only remote consoles, following the trend to unattended operation. We shall again dismiss or ignore the performance problems of console devices only because such problems are rare and of little general interest. The system service address space (CONSOLE) that drives consoles runs at a very high dispatching priority; console performance problems are either a symptom of a very sick system or indicative of a hardware or configuration problem.

- *Communications devices.* Terminals, printers attached to terminal control units, and devices masquerading as these are in this class. Many people regard the terminals they work with as "the computer." In many systems, response time at the terminal is the essence of the service level agreement. We will touch on only a few aspects of communications devices in this book, since our focus is on MVS, not networks. We will consider MVS's telecommunications access methods (VTAM and TCP/IP) and the ways in which their problems affecting response time may be managed. Problems affecting the portions of the network outboard of the processor complex are beyond the scope of this book.

- *Secondary storage devices.* Here resides the data[9] that justifies the term *data processing.* We call direct access storage devices (DASD or disks) and tapes "secondary storage" to distinguish them from processor storage, but the terms are usually implicit. Optical and magneto-optical storage devices are also in this category. When a system programmer complains about a "storage shortage," processor storage is usually

9 "Data" will be considered a singular collective noun. If there is a reason to consider an elementary item, "datum" will be used.

meant. When an MIS manager uses the term, it is almost always secondary storage that is in short supply. (It was easier in the old days, when processor storage was called "memory.") Our consideration of the I/O subsystem focuses on secondary storage for several reasons:

- It's where the money is (Willie Sutton's Rule). In many installations, the "DASD farm" is comparable in cost to the processor complex. Because each device is rather expensive but unspectacular, it is often more difficult to upgrade secondary storage than to get a new CPU.
- It's where the data is. The value of what resides on secondary storage far exceeds the value of the medium or the supporting hardware. It's not expensive to acquire raw computing power. Today's personal computers can serve individual users for prices comparable to those of "dumb" terminals. Why, then, connect them to a large central system? The data is the reason. The value of a centralized system derives from the value added by the sharing and centralized management of the data resource.
- It's where the problems are. Because data has value depending in part on the complexity of its use, conflicts are inevitable. Picture an enterprise in which all employees and many customers are online to all of the business's data. The need for controls and discipline is apparent. Problems that can arise are those of access control, integrity, and performance. We will examine the resulting performance problems in detail in later chapters.
- It's where the labor is. The next manifestation of a "black hole" for human labor—akin to "every person a telephone operator," solved in the early decades of the 20th century with the introduction of automated telephone exchanges—is "everybody into storage management." Storage management is an error-prone, under-recognized, and labor-intensive business. Until "system-managed storage" with DFSMS in MVS/ESA matures into a comprehensive automated system, the need for people and data processing efforts to be devoted to managing the data resource will continue to grow with the size, complexity, and importance of the resource itself.

- *Auxiliary storage devices.* Part of secondary storage is set aside for system purposes. Examples include the devices housing page data sets and the JES SPOOL and checkpoint data sets. Because problems surrounding these devices, particularly those used for paging, can have profound effects on system performance, we consider auxiliary storage apart from the rest of secondary storage.

3.3.1. IBM's I/O architecture

In systems that can run MVS, several different hardware elements together constitute the I/O subsystem. We examine each of them in turn, from the CPU outward to the device.

CPUs

In System/360, I/O involved the CPU a great deal. The original START I/O (SIO) instruction, for instance, did not *release* the CPU (complete execution) until the path through the channel, control unit, device controller, and device had been secured. Only then could an independent channel operate concurrently with the CPU. As CPU speeds grew rapidly, with I/O speeds trailing behind, relatively more CPU time would be spent waiting for I/O connections to be established. This same pattern was repeated at the channel level.

Beginning with the replacement of the START I/O instruction by START I/O FAST [RELEASE] in System/370, the trend of development from System/360 through System/370 to XA and ESA was to push I/O activity outward from the CPU and the channels, tying up hardware in the processor complex only when it was necessary for information transfer.

In Extended Architecture and Enterprise Systems Architecture, the CPU's role in initiating I/O is diminished, and the CPU impact of I/O completion is reduced as well. I/O path selection is done by the channel subsystem, so a complex section of MVS's I/O Supervisor (IOS) code is not needed. Restarting channel programs that have not been able to run is also a function of the channel subsystem, thus eliminating much I/O exception-handling code.

I/O interrupt control in XA and ESA has a finer structure than in System/370, so each CPU in a CEC can be disabled for such interrupts entirely or enabled only for selected classes of interrupts. In systems sustaining high I/O interrupt rates, a new instruction,

TEST PENDING INTERRUPT (TPI), allows a CPU to stay in I/O interrupt handling code to "pick off" the next candidate interrupt instead of allowing the interrupt to disrupt some other process.

The XA and ESA approach to the CPU's role in I/O setup is to "throw it over the wall" to the I/O subsystem. Upon completion of the I/O, a CPU is again involved to deal with the consequences of completion.

Channels and the Channel Subsystem

The invention of the I/O channel in the mid-1950s was one of the key developments that made today's operating systems possible. Before the time of the IBM 709, I/O was executed as an instruction in the CPU. The notion of *wait state* was equivalent to the idea of an idle system. With the I/O channel (and later with several channels), the formerly longest-running instructions (the I/O instructions) were transformed into simple control operations to manage the channel. The channel did the I/O "dirty work." While I/O proceeded in the channel, the CPU did not have to wait for the I/O operation but could execute other instructions that were not dependent on the I/O in progress. When the overlapped instructions were in a different program, what we now call multiprogramming was the result.

A channel is an independent processor, often as powerful as a CPU. It has read/write access to the same [central] storage as the CPU. The channel responds to a small set of instructions that control its operation. One of those instructions, START I/O or START I/O FAST, directs the channel to perform a series of activities:

- Obtain from known places the address of the beginning of a channel program and the identity of the I/O device to be selected.
- Establish the connection to the device.
- Process the control words that constitute the channel program. These channel control words (CCWs) specify the operation to be performed (read, write, or nondata "control") and the address and length of the data or control information to be transferred.
- Handle exceptional conditions that might arise during execution of the channel program.

- Signal the CPU when the channel program has completed execution.

The details differ from System/360 through ESA. As processing power has dropped in cost and increased in speed, more of the detail work is done by the channels and less by the CPU. In turn, intelligence and function have migrated further outward—to the control units and device clusters.

During the transition from System/360 to System/370, channels were expensive and few in number, even on large systems. Functional enhancements such as *disconnected command chaining* (DCC), *rotational position sensing* (RPS), and *block multiplexing* provided ways to increase channel productivity. During operations that did not involve the transfer of data, the channel could be disconnected from the control unit.

The control unit and device, acting on their own, would complete the operation (such as a seek to a different cylinder) and then reconnect to the channel. While one or more devices were executing disconnected commands, another could use the same channel path to transfer data. Block multiplexing was the co-requisite feature needed to keep track of multiple disconnected high-speed operations.[10]

The throughput benefits of block multiplexing, DCC, and RPS were realized, but at the expense of considerable performance degradation for less-busy devices when the channel paths were kept busy by active devices. In XA and ESA, the portion of DASD I/O time potentially spent waiting for channels to reconnect is reduced by the dynamic path reconnect feature, which allows the completion of an I/O operation on any channel connected to the device.

In System/370, reconnection could take place only on the initially selected path, leading to significant delay when that path was kept busy by other devices. XA and ESA eliminate most path delay, with support for up to four data paths per device adding to the benefits of dynamic path reconnect.

10 System/360 channels were either selector channels, allowing only one channel program to proceed at a time, or byte multiplexor channels, interspersing the data transfer of low-speed devices byte by byte.

System/390 and ESCON

With the announcement of System/390 and the ESA/390 architecture, IBM introduced a new kind of channel, Enterprise System Connection, ESCON. Compared with bus-and-tag channels, ESCON channels use a different data transfer discipline (serial instead of parallel), a different connection medium (glass fiber instead of copper), a different form of energy to carry the signal (modulated light instead of electrical pulses), and different connection topology (hub-and-spoke through an ESCON Director rather than point-to-point). ESCON channels and their associated auxiliary devices make possible greatly extended distances between processor complexes and devices, and much higher data transfer rates than with bus-and-tag connections.

Already available are ESCON data transfer rates of 17 megabytes per second. Maximum ESCON distances (using ESCON Directors) vary by type of connection: up to 60 kilometers for channel-to-channel connections, 43 kilometers for IBM 3172 and 3174 communication controllers, 23 kilometers for some magnetic tape controllers, and 17 kilometers for some DASD control units.

These increased distances, compared with the limit of 400 feet (122 meters) for parallel channels in data streaming mode, make for nothing less than revolutionary changes in data center design and layout. Instead of being restricted to a single floor or adjacent stories of a single building, a sysplex with its "locally" attached devices can span a campus, a downtown area, or an entire metropolitan region.

Control Units

A control unit (also known as a *storage director* or sometimes a *storage controller* for disk devices) is needed to synchronize I/O operations between channel and device, to handle routine error recovery, to translate I/O requests conveyed in CCWs into orders that devices can execute, and to handle path selection and the establishment of data transfer connections. These functions were handled in the past by dedicated hardware with all logic hard-wired in the circuits, much like CPUs of the past. As microcoded CPUs and channels became the norm, so did microcoded control units.

The cost of semiconductor storage continued to decline in the 1980s, and it became possible for IBM to introduce a new class of

control unit for high-speed disk devices, one that had storage as well as logic. The first such device for general-purpose systems was the IBM 3880 with the speed matching buffer (SMB) feature. SMB made it possible to hold a track's worth of 3380 data in a storage director, allowing the 3380 with its 3.0-megabytes-per-second data transfer rate to function on standard System/370 channels rated at 1.5 megabytes per second. SMB was a transitional feature, needed only until 3-MB/s "data streaming" channels came into general use.

The next way in which storage was used in disk control units came in the initial generation of cache control units. In these devices the storage was used not just to buffer between differing device and channel speeds, but to enhance the performance of disk devices. The IBM 3880 Model 11 was optimized as a paging subsystem, permitting obsolescent 3350 devices to function with greatly reduced apparent rotational delay and at effective data transfer rates greater than that offered by the device. Its companion Model 13 enhanced the response time of 3380s used for general data.

The first-generation cached IBM 3880 Models 11 and 13 were superseded by the Models 21 and 23, offering more cache storage at lower cost per megabyte and other improvements. The next generation of cache devices arrived with the IBM 3990 Model 3, offering very large cache sizes as well as four-path support. The 3990-3 also performs a speed-matching function (opposite in sense to that of the Speed Matching Buffer) along with its cache function; data can be transferred to and from cache storage at 4.5 megabytes per second or more on channels supporting such data rates. It is likely that this approach will become the norm for increasing channel speeds. Device characteristics (other than average access time, gigabytes per cubic foot, square feet of floor space per gigabyte, and cost per gigabyte) are becoming less important than cache capacity and cache data transfer rate.

Device Controllers

Some devices require an additional level of control. Most disk devices, for instance, are organized into *strings*, with a *string controller* needed to mediate the connections between devices and control units, and to translate the native physical characteristics of the device into the architecturally specified

count-key-data (CKD) format used with all MVS-supported DASD. The controller's logic is shared among all the devices in a string, but is not associated with any single device in the string. A common misconception is easy to understand, however. Because the string controller is often packaged with the initial device or cluster of devices on a string, some special preference is incorrectly attributed to the "head of string" device.

Because the string controller is needed for the full duration of data transfer operations (including SEARCH), a bottleneck can develop at this place on the I/O path. Just as multiple channel connections to multiple storage directors reduce bottlenecks at the inboard part of the I/O path, so multiple connections between devices and controllers can reduce the obstacle at the outboard interface as well.

IBM was not the first with such connections; other manufacturers' versions of 3350s had a feature usually known as *dual port*, providing two independent data transfer paths per string of devices. IBM supported two data transfer paths per pair of devices[11] on its 3375, and later two transfer paths per cluster of four devices on its initial 3380 model AA4. IBM eventually provided true dual port capability on its second-generation 3380 models AD4 and AE4. IBM's third-generation 3380s, models AJ4 and AK4, extended connection capabilities to four paths when used with IBM 3990 Model 2 or 3 control units. The same kind of connection is supported on the successor to the 3380, the IBM 3390, as well as on the IBM 9340 DASD subsystem.

The 9340, first announced in 1991, improved upon older generations of DASD in several ways. Its 9343-D04 Storage Controller connects to four ESCON channels and contains 2 megabytes of buffer storage. A 9343-C04 is also buffered and connects to four parallel channels. The buffer is required to reconcile differing data transfer rates for the device and the channel, and to implement *nonsynchronous DASD*, a storage subsystem design previously available only with cached control units, in which successive steps in a series of I/O operations need not be accomplished in strict lockstep with the passage of specific data blocks and

11 Because device controllers are not sold apart from devices, the distinction is hard to maintain. In IBM DASD nomenclature, the "A-box" of a string (as in 3380-AJ4) contains the (first) device controller. "B-boxes" are pure storage modules. The "C" designation was used for 3350s with alternate controllers and was brought back for the low-entry 3380-CJ2. The second controller in the not-quite-dual port configuration of the IBM 3375 was a "D" model.

gaps on a spinning platter. Instead, the buffer allows connect time to the channel to be minimized by managing SEARCH as a [channel] disconnected operation between the storage controller and the device. It also buffers data transfer to and from the device, thus avoiding RPS miss.

Devices

There is no "standard" DASD in today's systems. IBM 3380s, now dropped from marketing and withdrawn from the IBM product line, were replaced by 3390s. The 3390 and the newer 9340 subsystem has been displaced in the IBM product line by yet newer devices such as RAMAC, providing RAID benefits along with 3390-compatible virtual geometry. Other devices based on small form factors, massive caching, RAID, and nonsynchronous protocol are available from other vendors including Storage Technology (Storage Tek or StorTek)[12], EMC, Hitachi (HDS), and Amdahl/Fujitsu. Freedom from device geometry considerations beckons as well, with implementations much like the Integrated Drive Electronics (IDE) designs now common in personal computers. A page-oriented device addressed by page number much like expanded storage is within the capabilities of today's technology.

However, device geometry is still with us, and we must consider it as a necessary part of performance management. All current devices share the same general track format. Data on a 3380 track, for instance, is organized into 32-byte cells. There are 1499 such cells to a track, excluding track control information. For each record on the track, 15 cells are used for the count field and its gap, and 7 cells for the gap preceding the optional key field. Twelve bytes of ECC (error-correcting code) are added to each key field, if present, and to each data field. Each of these areas occupies some number of full 32-byte cells. Simple calculations show that each unkeyed record has an overhead of 492 bytes. (IBM's 3390 and 9345 share a similar but not identical data format, with 34-byte cells as well as a larger aggregate for additional error correction. The number of cells per track is greater in the 3390 than in the 9345.)

12 On June 10, 1996, IBM and Storage Technology announced that IBM would henceforth market and distribute many of StorTek's products, changing the names and making them part of the RAMAC product line.

A consequence of this track layout is that small physical records (blocks) are extremely inefficient on any such device. Space utilization efficiency of 80 percent or more is attained only with blocks of at least 2000 bytes. Eighty-byte blocks, still common in many installations as a legacy of punched cards, use only about 14 percent of the maximum capacity of a 3380 track. Space inefficiency is also speed inefficiency; while each 80-byte record moves at 3 megabytes per second, the 83 records on a track can't be transferred in less than a full device rotation of 16.56 milliseconds. The maximum effective data transfer rate for a series of 80-byte physical records is thus just 392K bytes per second.

The following table shows the characteristics of various direct access storage devices. Note that the effects of hardware compression, cache control units, or device or cache loading are not reflected in the table. A brief discussion of these considerations may be found in Chapter 9. Newer subsystems including IBM RAMAC, RAMAC II, RAMAC 3, EMC Symmetrix, and Storage Tek Iceberg (IBM RAMAC Virtual Array) and Kodiak (IBM RAMAC Scalable Array) also are not included in the table since these devices are so heavily dependent on caching and virtual device mapping. All of these devices outperform the older ones, but the physical characteristics of the base devices become nearly irrelevant when subsystem storage and logic add so much value to the basic physical configuration.

DASD Device Characteristics

Device Type →	3390-1	3390-3	9345-2
bytes per track	56,664	56,664	46,456
tracks per cylinder	15	15	15
cylinders per device	1113	3339	2156
MB per actuator	946	2838	1502
uncached transfer, MB/s	4.2	4.2	4.4
rotation time, ms	14.2	14.2	11.2
latency, ms	7.1	7.1	5.6
minimum seek, ms	1.5	1.5	1.5
average seek, ms	9.5	15	11
maximum seek, ms	18	33	20

DASD Device Characteristics			
Device Type →	3390-1	3390-3	9345-2
maximum paths	4	4	4
80-byte blocks/track	78	78	67
800-byte blocks/track	54	54	45
4096-byte blocks/track	12	12	10
8192-byte blocks/track	6	6	5
16,384-byte blocks/track	3	3	2

There is a change in packing efficiency across device types as block size increases. The 3380 had an advantage over the 3390 for small block sizes, whereas the 3390 passes the 3380 at 800 bytes and the 9345, even with a smaller gross track size, pulls even with the 3380 at the very important 4K block size.

3.3.2. Modeling I/O operations

Understanding I/O operations can be aided through modeling. An analytic model of a typical I/O operation for a conventional device (not an advanced storage processor) is relatively easy to construct, given the published facts about devices and their connections to systems. The technique is described in Chapter 9.

3.4. Summary

We have reviewed the physical resources to be managed by MVS, with particular emphasis on CPUs or processors, real storage (both central and expanded), and DASD input/output resources. In the next chapter, we'll look at how MVS organizes these resources into objects and structures that it will then manage.

3.5. Chapter Questions

1. Review the hardware configuration of your MVS system[s]. Trace its evolution over the past two years. Has the balance between CPU power and storage size improved or declined in that time?
2. How has the availability of logical partitioning affected your installation? Did your assessment of the CPU-storage

balance in your answer to question 1 reflect the full extent to which your installation uses or will use logical partitioning? Does your answer change with this consideration added?

3. If you were going to plan a data center installation today, how would the availability of ESCON channels affect its layout and its disaster recovery plan and arrangements?
4. Do you have any chronic I/O response time problems in your installation? Based on the device characteristics presented in this chapter, is there a simple hardware solution to this kind of problem? What would it take in terms of cost and inconvenience to improve system performance? Can you balance that cost against potential savings?
5. Estimate the amount of DASD space wasted in your installation because of block sizes that are too low. Can you relate low block sizes to performance problems as well?

Chapter

4

Virtual and Logical Resources

Things are seldom what they seem
Skim milk masquerades as cream

—W. S. Gilbert

Little Buttercup must have been an MVS designer. In MVS systems, things are often not what they seem. Let's consider a simple assembly language program that will read a sequential text data set, change all occurrences of the characters "extended storage" to "expanded storage," and write it back in place. As this program executes, the objects it deals with are hidden from direct view by a number of control functions, translations, and indirect references.

We refer to the direct translations or mappings as virtual counterparts of real objects, while the control functions, composite structures, and indirect namings define logical resources. These are not firm distinctions or standard terms. They may prove useful in understanding the pervasive nature of renaming in MVS.

4.1. Virtual Resources

In our hypothetical program, the initial need is to acquire a quantity of storage for buffers, work areas, control information, and save areas for general registers and other status informa-

tion. The storage is virtual and is acquired from the private area of the current address space.[1] If, say, 64K bytes is required, a *GETMAIN* or *STORAGE* macro-instruction in the source program is translated by the assembler to several instructions that place parameters in registers and then issue the SUPERVISOR CALL (SVC) instruction to invoke the appropriate service routine of MVS. That routine, part of the Virtual Storage Manager component of MVS, will allocate a 64K block of virtual storage from the private area and return its starting address to the program.

No *real* resources have been given to the program yet. The first instruction in the program that attempts to store data in the acquired storage will cause an *address-translation exception*,[2] a type of hardware-defined program check. The *page fault interrupt* caused by that exception will eventually be passed back to the Real Storage Manager of MVS, so that a page frame (real) in central storage, initialized to binary zeros, may be assigned to the page (virtual). This process will occur up to 16 times as the program's instructions are executed and "touch" (make reference to) each page in the acquired space.

All of this activity is transparent to the programmer and not important to an understanding of the program's logic. However, it is clearly relevant if we wish to understand the performance of the program and of the system in which the program runs.

We'll return to this example when we examine logical resources in the second part of this chapter.

4.1.1. Calling something by another name

Resources in MVS are made virtual ("virtualized") so that the same set of objects and addresses may be used in many address spaces concurrently, independent of the physical resources that will ultimately be used. Another benefit of virtualization is that

1 The address space is little more than what the name denotes: a linear space of contiguous byte addresses in which a unit of MVS work executes. An essential attribute of an address space is its size, the total amount of storage that may be allocated implicitly (as by causing a program to be loaded) or explicitly, by requesting storage through the GETMAIN or STORAGE service. Each address space is divided into a common area, shared with all other address spaces, and a private area, unique for each address space.

2 An attempt to read data from this area of storage would make no sense; nothing has been put there yet. If such an attempt were to be made, the same page fault would take place. In this case, servicing the page fault causes a central storage frame full of binary zeros to be assigned to the page.

the virtual resources can have standardized attributes (such as the range of usable addresses), regardless of the amount of physical resource available. Because most workloads other than batch jobs do not demand service continuously, most virtual resources are not needed full-time. Thus a scarce real resource may be shared among many virtual entities, with performance often approaching that of a 100 percent real system. This benefit becomes very significant when the virtual resource is of much greater aggregate size than the supporting real resource.

Storage is the best known of these virtualized resources, but external data may be made virtual using the *Virtual Input-Output* (VIO) service or through the exploitation of hiperspaces in MVS/ESA for storage of performance-critical data objects directly in expanded storage.

IBM's 3850 Mass Storage Subsystem (MSS) was installed in many early MVS systems. The virtualized entity in the case of MSS was an entire IBM 3330 or 3330-11 device, simulated with a combination of a 3330 or 3350 real device and the data contained on one or more strips of magnetic tape, rolled up in cartridges when idle, stored in a honeycomb-like repository, and brought to the read-write mechanism by robotic arms. This kind of virtualization has recently reappeared in two forms: the IBM RAMAC Virtual Storage Array (formerly known as Storage Tek's Iceberg) virtualizes disk volumes in a large array of small disks; the IBM Magstar 3494 Virtual Tape Server virtualizes tape volumes using a combination of physical tapes and intermediate disk storage.

With logical partitioning implemented by IBM's PR/SM feature or Hitachi Data Systems' (HDS) MLPF (or as separate domains with Amdahl's Multiple Domain Facility), whole systems are virtualized. With PR/SM in LPAR (logically partitioned) mode, a single CPU complex (or "side" of a multiprocessor) can look like as many as ten separate virtual instances in the 711-based ES/9000 9021s, with each CPU in a multi-engine configuration having up to ten "logical" appearances.[3] Each virtual instance or system is independently initialized (IPL'd) and has a defined

3 The treatment of CPUs in PR/SM, MLPF, and MDF is a classical *virtual* treatment, but IBM and HDS choose to call the virtual CPUs *logical partitions.* Amdahl calls them *domains.* The distinction may be useful; use of these terms helps to distinguish these hardware virtualizations from those provided with software assistance by VM.

complement of central storage, optional expanded storage, and channel paths.

IBM's Virtual Machine (VM) family of operating systems virtualizes systems through software—also making use of specialized hardware features that were forerunners of PR/SM, or of PR/SM itself—in the various levels of System/370 or System/390.[4] Each user of a VM system can be presented with the appearance of a complete System/360, System/370, or System/390 hardware configuration, ranging from a basic model to an ESA system, depending on the VM variant and the requested options. Although VM systems, most notably VM/ESA, can support "guest" operating systems, including MVS/ESA, within virtual machines, the typical VM end user does not start up arbitrary systems, but uses program development tools and other application interfaces built on CMS (Conversational Monitor System[5]) as a time-sharing user.

4.2. Physical backing of virtual resources

Virtual resources are useful abstract entities, but to get work done they must be backed up by appropriate real resources at the time of use. It is a major purpose of operating systems such as MVS to manage the real resources and to control assignment of those resources to the virtual objects and work units that the operating system supports.

4.2.1. Virtual storage and the paging subsystem

Because the aggregate size of virtual storage in a typical MVS system is very large compared with the size of the real storage resource, the data in virtual storage pages must be stored somewhere else when real (central) storage frames are not available. A moderately large system might have 800 active address spaces, requiring an average of about 175 pages each. The 140,000 page frames (560 megabytes) needed to "map" all of this virtual storage to real storage are not found in any but very large systems. Even when that much storage is available,

4 VM/ESA uses the PR/SM hardware to support multiple preferred virtual guest operating systems; such use is mutually exclusive with LPAR mode.

5 This incarnation of CMS is based distantly on the original Cambridge Monitor System, a simple single-user operating system very similar to PC-DOS in power and ease of use. CP/67, the forerunner of VM/370 and VM/ESA, was originally intended to be a simple "hypervisor" providing multiple CMS virtual machines to time-sharing users.

it would be extravagant to install it just to support such a workload. Experience shows that 80 to 90 percent of interactive users are idle at any given time. Real storage of 30,000 to 40,000 frames should be sufficient to support the workload with acceptable response time.

Pages not currently mapped to real storage must be kept elsewhere. In systems with expanded storage, processor storage is divided into *central storage*, similar in concept to the older undivided real storage, in which instructions and data may be accessed by the CPU, and *expanded storage*, which holds data not accessible to the CPU except as page-sized objects to be exchanged to and from frames of central storage. In such systems, expanded storage is the first choice for storage of pages not currently needed by the CPU but likely to be needed soon.

Normally each frame of real (central) storage may be mapped to a virtual page of one address space's private area or to a page of the common area, or be kept on the *available frame queue* as a candidate for future assignment. However, a facility introduced in MVS/ESA SP 5.2 allows a page in any set of private areas to be mapped to the same frame. This facility, originally known as *captured storage*, was later renamed as the *shared page* facility. Using shared pages, common information may appear in many address spaces at (possibly) different virtual addresses using only one set of page frames. If a program changes the content of a shared page, the change is captured on a new frame in the address space making the change. The shared page is not changed.

With or without expanded storage, all pages of all virtual storage in an MVS system must at least potentially be backed by *slots* of *auxiliary storage.* Each slot is the size of a page (4096 bytes), and MVS's Auxiliary Storage Manager manages the assignment of pages to slots and initiates the input/output operations needed to move pages between central and auxiliary storage. This resource is organized into *page* (or *paging*) *data sets.* Collectively, the set of page data sets, along with any optional *swap data sets*, is known as the *paging subsystem.*

Before large real storage sizes (and the use of expanded storage) became common in MVS systems, the paging subsystem was often the single most critical performance management challenge in MVS. Factors to be considered included number, size, and placement of page data sets, the degree of interference from and to other I/O, the use (and then the number) of swap data sets,

the size of real storage, and the interaction of MVS's internal workload management controls with paging performance.

IBM speakers at user-group meetings would often suggest that the days of storage constraint were at an end, and that tuning the paging subsystem may be of even less importance in the future. Acknowledging that there is now a trend away from real storage constraint, nevertheless we recognize in this book that not all systems are the latest, that not all budgets are the most generous, and that those who pay for and use MVS systems are very ingenious in devising ways to exploit the real storage resource. We also note that OS/390 has the capability of enabling a new generation of applications implemented in a storage-rich environment. Whatever temporary ease there may be in real storage may well become constraint again as those new applications come into full production and maturity. In short, we will assume that optimizing paging subsystem performance to control paging delay remains a worthwhile concern.

Virtual Storage in MVS/370

We begin looking at virtual storage by considering MVS/370, obsolete though it is. The layout of MVS/XA and MVS/ESA is easier to understand with that background.

The layout of virtual storage in an MVS/370 system is shown in Figure 4-1. Each address space is 16 megabytes and includes all of the MVS *nucleus, common storage area* (CSA), *link pack area* (LPA), and *system queue area* (SQA). The remaining space, below a 64-kilobyte *segment boundary* defining the limit of the *system area* and above the segment boundary defining the nucleus, is the *private area*. From that area is deducted the *local system queue area* (LSQA), containing control information (such as *page-translation tables*) required by the address space.

What is left is the available area for the real business of the address space. This area was originally to have been guaranteed to be no less than 8 megabytes. In many MVS/370 systems, it was less than 6 megabytes. For a program package to be portable across most MVS/370 systems, it must fit into the smallest available private area in any of those systems. The phenomenon of limited virtual storage leading to program design compromises was dubbed *"virtual storage constraint."* Successful efforts to roll back the system area boundary and make more private

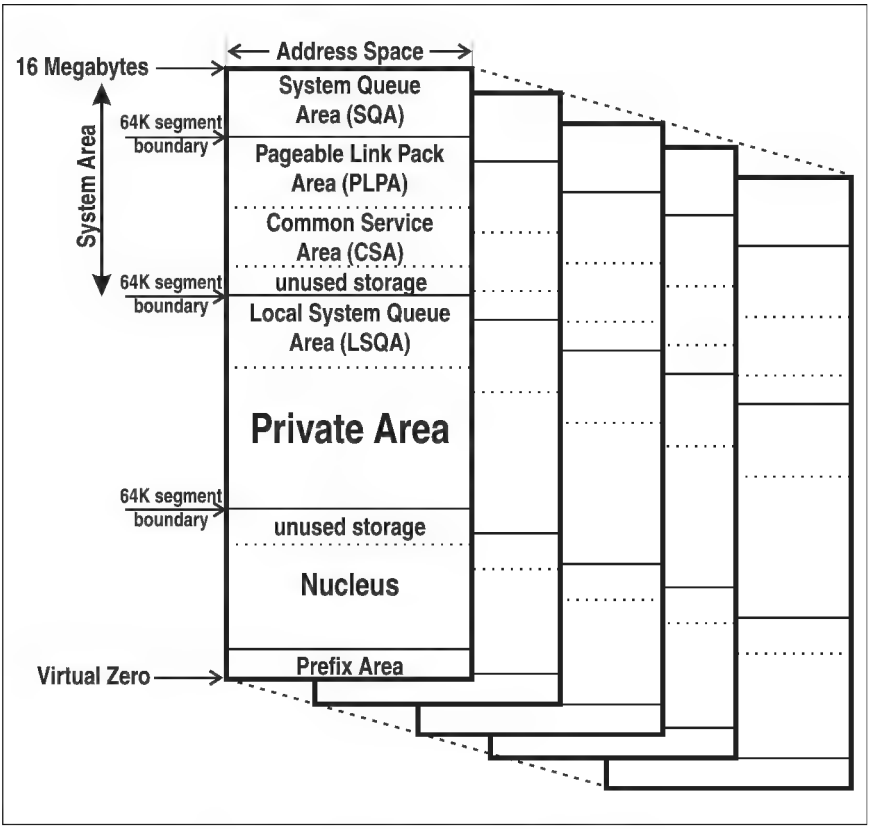


Figure 4-1. MVS/370 Virtual Storage Layout.

area available were and continue to be announced by IBM as *virtual storage constraint relief* or VSCR.

Splitting Large Programs Across Address Spaces. Many programs, particularly subsystems like IBM's Customer Information Control System (CICS) and Information Management System (IMS), frequently cannot fit in the limited confines of a single address space. The MVS versions of IMS (IMS/VS) are designed to use at least two address spaces, a Control Region and one or more Message Processing Regions.[6] More recent versions of IMS can use additional address spaces to manage data base I/O and recovery management functions.

6 The "region" designation is a leftover from MVT and SVS. Each such region is actually an independent address space.

CICS is less formally structured than IMS.[7] Multiple address spaces in CICS may each perform a full range of functions with little or no intercommunication, or they may be organized into specialized terminal-owning, resource-owning, and application-owning address spaces. Again harking back to MVT terminology, the structuring of CICS into multiple cooperating address spaces is known as Multiple Region Operation, or MRO.

Inter-address Space Communication. Programs that communicate across multiple address spaces do so through a variety of means. The earliest one used, before *cross-memory services* (CMS)[8] became available in hardware, was an indirect discipline involving the scheduling of dispatchable work units to run in the target address spaces. These work units, *service request blocks* (SRBs), are preferred by the dispatcher above all tasks in the address space when they are selected for dispatching.[9] The task in the address space that asked for the service needs to wait for the SRB's completion, often signaled by a complementary SRB scheduled in the caller's address space by the server.

Asynchronous cross-memory communication via SRBs has several drawbacks. The use of SRBs involves extra overhead and potential delay. The *SCHEDULE* macro's SUPERVISOR CALL (used to create SRBs) can be issued only by an address space running in authorized state, and any data required for communication needs to be passed in common storage so that both communicating address spaces can have addressability to it. This use of CSA contributes to the very virtual storage constraint that the communication across address spaces attempts to relieve.

7 The informal structure of CICS appears to be becoming less so in the MVS/ESA releases of CICS since Version 3. These versions exploit MVS/ESA features such as data spaces, make use of the MVS dispatcher, and incorporate multitasking internal structures to ensure that they can take full advantage of multiprocessor configurations.

8 This CMS has nothing to do with the CMS of VM/370. Some writers prefer to use "XMS" to denote cross-memory services, but this choice has been pre-empted; "CMS" has been institutionalized as an MVS lock name. The term "cross-memory" itself is another throwback to MVS prehistory. Because the original name of MVS was MVM, for Multiple Virtual Memories, terms like "memory create" and "cross-memory" are deeply ingrained in the internal literature of MVS (for instance, the comments embedded within the source code) instead of more accurate and less confusing terms referring to address spaces.

9 At least they were prior to SP 5.2. A new class of dispatchable unit called *client SRB*s were introduced then, providing the opportunity to run service code with the low overhead of SRBs, but with the dispatching priority of the invoking task. Also in that release, some SRBs become preemptible, thus allowing more opportunity for the dispatcher to choose waiting work units.

With the support of CMS hardware by MVS/SP Version 1, a much more efficient means of communication became available. After required control structures had been set up, an address space could transfer control synchronously to another address space, resuming operation following the synchronous return. No supervisor services are needed after initialization. Data can be moved directly between address spaces known to each other, and the address spaces need not run in the authorized state.

The "horizontal splitting" made possible and efficient by using CMS helped MVS/370 to remain viable long after its 16-megabyte virtual addressing capability had become inadequate.

4.2.2. Virtual storage in MVS/XA and MVS/ESA

The virtual storage layout of MVS/XA and MVS/ESA is shown in Figure 4-2. Note that data spaces, described below, are available only in ESA.

A major objective of System/370 Extended Architecture (XA) was substantial VSCR. Certainly this was a possibility; the virtual address range of XA is 128 times as large as that of MVS/370. However, many key subsystems did not make use of the private area above the 16-megabyte "line" for years after the original release of MVS/XA. VSCR came from the operating system itself as more system address spaces were split out of nucleus functions. The historical details are provided in Chapter 1.

Another part of XA's VSCR was the system's use of the "extended" areas "above the line." As soon as major parts of the operating system could be rewritten to exploit 31-bit addressing and reside above 16 megabytes, those parts could be moved from Nucleus to Extended Nucleus, or from PLPA to EPLPA. Because of the long history of MVS, which has led to many dependencies on system data remaining where it had "always" been, and the important goal of release-to-release compatibility, this was a protracted process and still continues in MVS/ESA.

In Enterprise Systems Architecture (ESA), another hardware change makes new forms of addressing growth possible. Sixteen *access registers* (ARs) correspond to the 16 general registers. For each general register used as a base register,[10] a new addressing mode uses its paired access register to supply indirectly a

10 Access register 0 has no defined use as yet, since general-purpose register 0 cannot be a base register.

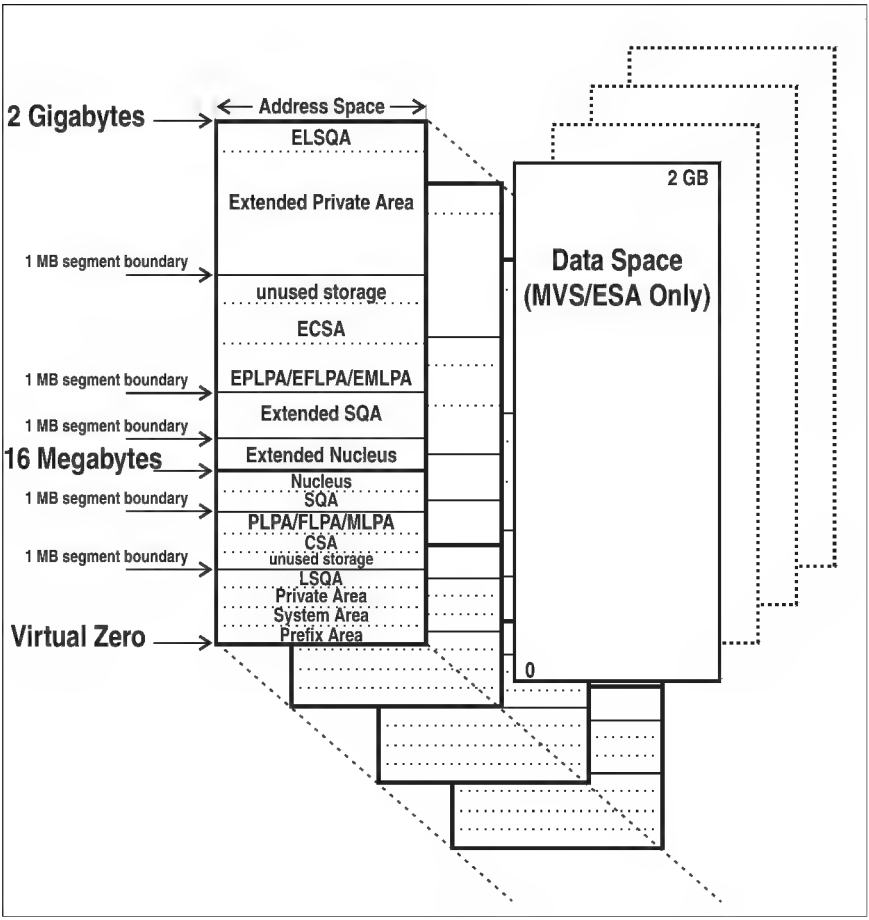


Figure 4-2. MVS/XA or MVS/ESA Virtual Storage Layout.

pointer to an individual segment table origin. Thus data (but not executable instructions) may be found in a separate address space. That address space may be specified as a data-only address space, or *data space*. A data space's segment table does not include the common area's segments, so it provides a full 2-gigabyte address span for private data, and up to 15 of them (one per access register other than AR0) may be active in support of an address space at any time.

Using data spaces, subsystems may be restructured to gain new levels of performance (by minimizing the need to store ac-

tive data on external devices) and integrity (by separating code from data).

Hiperspaces in MVS/ESA

Another mechanism in MVS/ESA to save input/output operations is the *hi*gh-*per*formance *space*, or *hiperspace*.[11] Note that hiperspaces are *not* shown on the virtual storage layout illustration. Hiperspaces reside outside the virtual addressing structure, although each hiperspace is visualized as a byte space of up to 2 gigabytes in size, but with an addressing granularity of a page (4096 bytes). The data within a hiperspace may be examined or manipulated only when it is brought into an address space or a data space by operations analogous to data set *READ* and *WRITE*.

The performance advantage of a hiperspace is achieved by materializing a "data set" in it and letting the paging subsystem do the I/O for only those portions that are read or written. A high-performance type of hiperspace is defined to reside only in expanded storage, and is called ESO, for Expanded Storage Only. More mundane hiperspaces may reside in a combination of expanded and auxiliary storage. Hiperspaces never reside in central storage. For response-critical applications requiring very large data structures with low access density, hiperspaces can provide nearly unlimited definable storage. Unlike data spaces, which are limited to 15 active at any time by the number of base/access register pairs, the number of simultaneously active hiperspaces is limited only by address space limits in the *SMF* exit IEFUSI. The maximum combined size of data spaces and hiperspaces owned within an address space is $2^{24} - 1$ *megabytes;* the maximum number of such objects is $2^{32} - 1$. The IEFUSI exit may be used to set limits lower than these defaults.

Since application programs need to be conceived, designed, and implemented to take advantage of such potential, it may be some time before the potential of hiperspaces is fully realized. More likely, subsystems such as the MVS/ESA Storage Management Subsystem (SMS) will take the lead by using hiperspaces to automatically stage performance-critical data sets to expanded storage at the time of need. This is already done for data

11 Hiperspaces do not depend on access registers or on any other unique hardware features of ESA. They are an example of system improvements implemented only in ESA since IBM has withdrawn support of XA.

sets defined as *Partitioned Data Set–Extended*, or PDSEs. When the MSR (milliseconds response) of the SMS storage class defining a PDSE is of a value indicating the *always-cache* category, up to 256 megabytes of hiperspace is used as the working copy of the data set. (PDSE directories are always maintained in data spaces.)

4.3. Logical Resources

We can now return to the example that opened this chapter. We have a simple program to assemble, link-edit, and execute. We'll just submit it as a batch job through TSO and wait for the results....

Uh-oh! Unfortunately we have overlooked an essential part of the picture in blithely assuming that our example program has begun executing as if by magic. We must dig deeper to understand first, how the job gets "into" MVS and eligible for execution; and second, how the MVS dispatcher gets an opportunity to dispatch the work unit in which the program will run and how it decides which address space to dispatch whenever it gains control.

4.3.1. Logical storage resources

As described in the next chapter, the job entry subsystem (JES2 or JES3) controls access to the set of executing jobs by means of a set of queues called *initiators*. The initiators are started and stopped through operator commands, although automation facilities are often used to issue those commands based on schedules or demand. Each initiator serves one or more *job classes*, and the job class of a job is specified in the JCL for the job, subject to amendment that the JES may exert through program extensions called *exit*s.

The process by which the executable work unit begins execution is a bit complex, but let's concentrate on the logical resources that stand in the way of getting the work done. Once a job gets though the JES queue and is ready to be started, it passes through a classification process, usually on the basis of the job class, that assigns it to a *performance group* (if the system is not in Workload Manager goal mode) or to a *service class* in the case of running in goal mode. Performance groups and service classes are logical resources based on similar need for resources or simi-

lar need for service, respectively. The discussion must split here between the approaches taken in connection with these two logical resources.

Access to Storage Resources in Compatibility Mode

Compatibility mode is shorthand for "not goal mode," covering the operation of the System Resources Manager (SRM) in every release since SE2 in 1978 through the current versions in the specifically chosen compatibility mode. In compatibility mode, there is a logical resource that controls access to storage, multiprogramming level (MPL).

MPL management. MPL is controlled in turn at two levels. At the global level, the system determines through a series of parameter evaluations whether the current overall MPL[12] is optimal or requires adjustment upward or downward. Below that level, the MPL is distributed through another class of logical resource, domains. Each performance group may be divided into operational periods in which the specific operational parameters for a work unit are provided based on the amount of resources already consumed by the work unit. One of those parameters is a number identifying a specific domain.

The domain itself can be shared across multiple periods of multiple performance groups and is a logical resource controlling access to a subset of the multiprogramming set. Prior to SP 4.2 the MPL within a domain was made up of swappable address spaces only, with an option to include nonswappables in the count. As of 4.2 nonswappables are always included, with no option to do otherwise. We'll consider the SP 4.3 treatment only.

It becomes obvious at this point that the level of abstraction is very high. If we ask the question "Can this job be allowed to run or will it be swapped out immediately?"[13] we must pass through the following steps:

- it is classified into a performance group
- the domain specified in the first or only period of the performance group is identified

12 The set of address spaces comprising the overall MPL is known as the multiprogramming set.

13 Even if the work unit is to be made nonswappable, it starts off as swappable.

- the in-target of the domain is compared with the number of swapped-in address spaces in the domain
- if the in-target exceeds the current MPL, the address space is allowed to execute, otherwise it will be swapped out

The target MPL for a domain as of SP 4.3 varies between specified upper and lower limits, controlled through an intermediate variable, contention index, as determined by an additional parameter based on service rate.[14] There is usually a gap between the current MPL and the in-target, allowing an additional address space to begin execution. However, careless specification of domain MPL constraints (*e.g.*, by not accounting for nonswappable address spaces in systems beginning with SP 4.2, or by setting unrealistically low upper constraints) can close the gap and force undesirable swap-outs. The details may be found in Appendix B.

Access to Storage Resources in Goal Mode

The domain structure, even though simplified in SP 4.2 and further adjusted in 4.3, proved to be too abstract and too error-prone as a means of obtaining optimum performance for key workloads. In Workload Manager goal mode the structure is swept aside completely. Instead, control of access to the storage resource is used to manage work in service classes according to their goals. MPL is distributed to service class periods as it was to domains, but Workload Manager hides the process and does not materialize the MPL logical resource. The adjustment of the global MPL continues to be based on measurable values like the global UIC, but the logical resource of MPL is not a visible part of goal mode.

4.3.2. Logical CPU resources

Once our little job has been allowed to start and is swapped in, it is ready to receive CPU service. We now look at how access to the CPU is controlled.

Dispatching Priority

Dispatching priority is the logical resource guarding access to the CPU. How are the dispatching priorities established and how does the dispatcher use them?

14 A fixed contention index is another option.

This has been one of the busiest areas of evolution in MVS. In more innocent times at the dawn of the OS/360 area, it was very simple. The JOB or EXEC statement allowed the specification of priority through a simple parameter: **DPRTY=(n_1,n_2)**. The numerical values of n_1 and n_2 range from 0 through 15, with **(15,15)** representing the highest possible priority and **(0,0)** the lowest. The two-part specification mapped to a single priority value ranging from 0 to 255, as $(n_1 \times 15) + n_2$. Since in those early days jobs were handed in as decks of cards, it was easy for an operator or production control person to screen the JCL and change those priorities that appeared to be unjustifiably high.

As TSO and automated job-submission tools began to take hold, this system broke down. A job could be submitted directly to the job queue and no one would see it until it began execution. Some external means of controlling dispatching priorities was needed, and the earlier implementations of SRM began to address the problem.

SRM Priorities from SE2 to SP 4.3

We'll skip over some details of evolution and look at the final result prior to MVS Version 5. The SRM from 1978 to 1994 (and Version 5 and OS/390 in SRM Compatibility Mode) allowed the assignment of dispatching priorities through a mapping from the basic 0–255 range to a subset, split into bands and subdivided within each band. Instead of the priorities being subject to contention among users *via* JCL, they could be controlled centrally through parameters specified by system programmers in their role as system administrators. Work units (jobs, TSO sessions, and started tasks) could be classified according to various criteria to *performance group*s which in turn contained priority specifications. The details of the priority specifications and other aspects of the earlier SRM may be found in Appendixes A and B.

Problems in Dispatching Priority Control

Each time the MVS dispatcher runs, it follows a simple rule: dispatch the highest priority ready task of the highest priority ready address space. The rule sounds fair, but it begs the question of how the queue order is determined. Dispatching priority has 255 distinct levels; in practice a maximum of 160 levels is

managed by SRM but those 160 come down to only 70 distinct specifications.

Because of the limited number of priority choices, there will usually be some number of address spaces of equal nominal priority. A means is needed first to resolve the question of which address space of those at equal priority should next be dispatched. A series of evolutionary changes have led to the latest implementation, called the *fair access* algorithm, which seeks to equalize the selection opportunity among address spaces of equal priority.[15] In other words, something like the old "rotate" algorithm is back, but it is not externally visible.

The larger question remains: how is priority determined? Very simply, SRM (in compatibility mode) does what you tell it to do. The carefully layed out numerical dispatching priorities translate to a queue order, usually with subqueues of equals at several positions in the main queue.. With very little adjustment, the order is static but the various address spaces cycle in and out of readiness to run. In MVS systems since 2.1.2, only the ready address spaces are visible to the dispatcher.

The problem lies in the rationale used to set up the priorities. It is classical resource management. The CPU resource is doled out according to the priority ordering in the logical space of dispatching priorities. However, the priorities are set up in order to manage the performance of workloads.

We now come upon the main deficiency of resource-oriented resource management: within MVS there is no measurement of workload performance and no feedback of workload performance deficiencies to the mechanism responsible for setting up dispatching priorities. The measurement and feedback, when done at all, are external to the operating system, the former by various performance monitoring programs, the latter through human intervention.

Workload-oriented management of resource allocation in MVS prior to version 5 is indirect, labor-intensive, and uncertain. If a workload is not performing adequately and the business priorities of the installation determine that improvement should be attempted, it is first necessary to determine what will help. Sim-

15 With the advent of dispatcher enclaves as work units independent of address spaces, the dispatcher now seeks to provide fair access among work units rather than address spaces.

ply increasing that workload's dispatching priority may not help if CPU-access delay is not a significant problem—and doing so may introduce delay for other workloads.

Many who attempted such tuning became wary of change and acted only when the most severe problems appeared. A change in methodology was necessary for many reasons.

Dispatching Priority in MVS/ESA Version 5 and OS/390

With the arrival of Workload Manager goal mode in MVS Version 5, the opportunity exists to replace resource-oriented resource management with workload-oriented resource management. Dispatching priority is no longer *externalized*, but it is used internally by the SRM as one tool in a set of tools that manage workload performance by controlling access to resources. In goal mode, the logical resource of dispatching priority no longer exists. However, the latest version of the dispatcher algorithms described above is retained in goal mode. Indeed, the changes were implemented in anticipation of goal mode.[16]

Our discussion of goal mode appears in Chapter 7.

4.3.3. Logical I/O resources

Going back once again to the example that introduced this chapter, our simple editing program has finally commenced execution. It has established its environment by acquiring its own working virtual storage; it has saved the caller's registers; and it is now ready to get to work. Remember, we were going to read a text file, look for a particular character string, replace the string with another of the same length, and write the changed information back to its original location.

To perform a data-editing operation that is very simple to do in a high-level language, our assembly language programmer must deal with a formidable series of structures interposed between the program and the data "out there" on a device.

Data recorded on a track of today's DASD, such as the IBM 3390, is written as a series of fixed-length cells, regardless of the data organization we see at a higher level. The cell size for 3380s is 32 bytes, and it is 34 bytes for 3390s or 9345s. The device is at its most efficient if all the cells on the track are written

16 For a comprehensive description of the dispatcher's evolution, see Bernard R. Pierce's CMG95 late-breaking paper, *Dispatching Management in MVS - TCBs to Enclaves*, in *CMG Transactions,* Issue 91, August 1996.

at once; only one instance of overhead is needed if only one *physical record* is written per track.

Physical record? What happened to cells?

The cells are not visible to the operating system. A virtual structure is superimposed on the track organization by microcode in the device, string controller, and storage director. It is at this level that the most recent storage processor subsystems and the SCSI-based inboard DASD of the Multiprise 2000 series, the P/390, and the R/390 meet the past systems. They all present the same virtualization of "conventional" devices as seen by the I/O subsystem. The virtual picture is that of a *count-key-data* (CKD) device, differing only in quantitative physical details from the first DASD on System/360. In the CKD architecture, each track is organized into physical records, and these records correspond to the data moved by a single READ or WRITE CCW. Thus the discussion on block size at the end of Chapter 3 really should have been preceded by a discussion of the virtualized data structure. However, it's easier to accept the notion of CKD as a hardware architecture and ignore the underlying details. When it's desirable to understand *why* the capacity falls off so dramatically at small block sizes in some implementations, the details become important.

Data Organization on DASD

MVS supports several logical organizations of the data contained in the physical records of I/O devices. Of interest in the current example is the *physical sequential* organization. In this type of *data set organization* (DSORG), the operating system, through an *access method*, recognizes that the data exists in order, and that a request for the "next" block or record does not require positioning information.

Data Sets

The aggregate of data which is commonly called a *file* in the jargon of other systems is known as a *data set*[17] in MVS. The data set in our example is a *sequential data set*, and our program will read and update it by means of the queued sequential access method, or QSAM.

17 This term is sometimes spelled as a single word, "dataset."

The data set is the fundamental unit of data management in MVS. It corresponds to a label entry (known as a *data set control block*, or DSCB) in the area of each disk volume that is reserved for such entries. Most permanent data sets are also *catalogued*;[18] such a data set can be found and retrieved by name only, without having to know which volume it is on.

Late Binding

The previous discussion may have seemed a bit vague, but it was meant to be. MVS has a deliberate amount of vagueness built in. In systems less sophisticated than MVS, a file might be found in a *directory* of a specific disk device, and all of its attributes would be either implicit or specified directly in its directory entry. A strength of MVS is the idea of *late binding*; data set attributes may be specified or amended at the time of use. This is accomplished by a multiple level of specification. In the case of a batch job, the chain of specification begins with the job control language for the job step in which the program will be executed. In turn, data set information is specified in the following layers:

- **DD Statement.** The DD, or data definition, JCL statement has a name specified in the program. (In the interactive TSO environment, an ALLOCATE command is the counterpart of the DD statement, and the term file name is used as a synonym for DD name as a tag to denote the program's view of the data set.) The idea of the DD was the key innovation of late binding. A program needs to have some way of identifying the data it will use or create, but if a data set name is embedded in a program, the program will have little general usefulness. The DD provides a layer of insulation. The program refers to a constant DD name, but the DD or ALLOCATE can name any data set meeting the needs of the program. The data set's attributes may be specified at the time of its creation, and some of those attributes can be amended at each time of use.

- **Data Set Control Block (DSCB).** For a pre-existing data set, the DSCB (a more complete version of the *file label* in earlier systems) records the attributes with which the data set was created or last updated. Some of these attributes (such as the

18 In the system-managed storage (SMS) environment of MVS/ESA, *all* permanent data sets are catalogued.

data set organization) cannot be overridden by DD specifications or from within the program; others, such as the physical record size (known in MVS as *block size*), can be overridden.

- **Data Control Block (DCB).** The DCB is both an MVS control block and the macro-instruction that generates it. In a program before it's compiled or assembled, the DCB contains, at minimum, the view of the data set required for the successful resolution by a compiler or assembler of references to it within the program. Those references to the DCB are as targets of I/O macros or corresponding higher-level statements within the program. In our example, we would expect these macros to be *OPEN, GET, PUTX*, and *CLOSE*.

- **OPEN Macro.** Once the program is loaded and executing, *OPEN*ing a DCB binds dynamically the program's view of a data set to the real data set. In the original concept of OS/360, *OPEN* would complete the fields of the DCB, dynamically load the appropriate *access method* modules, create or update the DSCB for a data set on a direct access device, acquire storage for buffers, and, for a sequential data set to be used for queued input or update, "prime" the buffers in anticipation of the first *GET* request.

This set of functions has been resilient over the years. The only major difference in an MVS/ESA system of today is that the access method code does not need to be physically loaded; it is likely to be already resident in the pageable link pack area (PLPA). As ESA's new capabilities are exploited, this too may come full circle. It may become more appropriate to load, on demand, some access method modules from libraries maintained in data spaces by the Library Lookaside Facility than to take up virtual storage for them in PLPA.

An even more flexible set of late-binding primitives was introduced as a matter of necessity when TSO was added to MVT. The added set of services is *dynamic allocation*.[19] With dynamic

19 MVS's *allocation* component handles the assignment of physical resources (devices) and logical resources (data sets) to address spaces. In OS/360, allocation was part of the *scheduler*, the set of services responsible for sequencing batch jobs through the system. With the arrival of TSO, the need for dynamic allocation forced changes in venerable OS/360 allocation code. We might view allocation today as a structure built on dynamic allocation as a basic set of primitives. Many current MVS subsystems use dynamic allocation rather than relying on static JOB step allocation.

allocation, a data set name and its attributes can be elicited interactively from the terminal user, placed in a DCB by invoking dynamic allocation services, and allocated to the TSO session. The DCB can then be *OPEN*ed, and the data set created, or otherwise manipulated, without the need for even a DD name as a fixed external reference.

SYSIN, SYSOUT, and Terminal Processing

Most batch jobs and many subsystems produce printed output. A DD statement as simple as

```
//SYSPRINT DD SYSOUT=A
```

and corresponding program statements such as

```
          ...
PUT       PRINTIT
...       ...
PRINTIT   DCB  ...DDNAME=SYSPRINT,...
...
```

direct the desired output to the job entry subsystem (JES; either JES2 or JES3) to place on its "Class A" queue for output. The details of how it will be printed are left to the JES. SYSOUT processing is an extreme example of late binding.

If the same program is run in TSO, the user probably won't want printed output, but would expect the program output to be displayed on the terminal. Without a program change, the TSO user need only enter a statement in *TSO Command Language* to create the association between the internal file name and the external data set, which in this case is a data stream directed to the terminal. The user would enter:

```
alloc f(sysprint) da(*) reus
```
[20]

or accept the same allocation, found by default in most TSO LOGON procedures, to direct SYSPRINT output to the terminal.

The job entry subsystem can manage input streams as well as output streams. Batch jobs can obtain control information or

20 In the command, **alloc** is an alias for **allocate**; **f** is the minimal abbreviation for **filename**; **da** is the minimal abbreviation for **dataset**; the * denotes the terminal; and **reus[e]** makes the allocation independent of whether the filename had been previously allocated. The last option is available only with IBM's TSO Extensions Licensed Program, now included as a standard element of OS/390. Note also that the entire command is case-insensitive.

low-volume data input from the JCL itself; the method of doing this is to use the "**DD ***" or "**DD DATA**" option:

```
//SYSIN DD DATA
```

This command will associate what follows that statement in the JCL data set with the SYSIN DD name. The input ends when the MVS reader-interpreter encounters an input record beginning with the characters "/*." The TSO counterpart for input from the terminal is:

```
alloc f(sysin) da(*) reus
```

4.3.4. Other logical resources in MVS

A data set OPENed for output is an example of a *serially reusable resource* (SRR), so named because many work units can use such a resource, but only one work unit can use it at a time. There are many other such resources in MVS. Depending on the nature of the SRR, various kinds of damage may result if several work units (tasks or SRBs) attempt to use one concurrently. A mechanism is needed to protect SRRs from such damage.

At least two such mechanisms are available. For SRRs that are used frequently and have fast service times, hardware-assisted *locks* are used. An example is the SALLOC lock, used to ensure that only a single process is allocating virtual storage at a time. Locks are generally used by "system programs," *i.e.*, those that call on or provide low-level services.

For more ordinary resources, the GRS (Global Resource Serialization) address space in MVS provides ENQ (enqueue) and DEQ (dequeue) services, invoked by similarly named macro-instructions and their generated SUPERVISOR CALLs. Like locks, ENQs and DEQs are usually concealed by higher-level services. When a data set is *OPEN*ed, for instance, the OPEN service routine will issue an *ENQ* against a major resource name of SYSDSN and a minor name identical to the 44-character data set name. The *ENQ* macro will specify the EXCLUSIVE attribute if the data set is to be used for output or update, or SHARED if the intent is input only. If a task requests an ENQ when another task has an EXCLUSIVE ENQ for the same resource, the requesting task will be notified of the condition in a return code or will wait until the resource is available.[21]

Contention for real resources is held off by the ENQ/DEQ mechanisms. Instead, contention is transferred to the logical objects

represented by ENQ resource names. When programs are sloppy in either securing or releasing ENQs, contention between address spaces can arise. One of the most frequently seen performance-killing conditions in the TSO environment is "ENQ lockout." Because it is possible to secure some but not all resources needed for a program, tasks that do not release ENQs if not all are available can "fight" with other tasks and cause system productivity to suffer. As we'll see in Chapter 9, performance monitors can detect such slowdowns and often offer a means to cancel the address space holding the blocking ENQ.

4.4. Summary

MVS operates through virtual and logical resources to:

- protect critical system information from damage
- facilitate resource sharing
- insulate system users and program designers from the complexity of the real system resources
- project an appearance of resources better suited to the needs of an operating system, its subsystems, and its workloads than the bare hardware can do

The cost of these protections and flexibilities is system overhead. The overhead is usually far outweighed by achieving the ability to make effective use of a vast array of system resources. The performance management challenge is to understand the nature of the resources and to be able to look beyond them to solve underlying problems.

4.5. Chapter Questions

1. Does your MVS system have a virtual storage constraint? Why? Create a no-cost plan to eliminate the constraint or show why that is not possible.
2. Determine how extensively central storage is overcommitted in your MVS/ESA system. How much is backed by expanded storage and how much by auxiliary storage?

21 Notice how easy it is to confuse the logical gatekeeper for a resource with the resource itself. The resource no longer may be in use by the other task but GRS will not know that until a DEQ is issued for the correct resource name.

3. Of the DASD problems in your system, which are caused by contention and which are resource exhaustion problems?
4. How do DASD resource management practices contribute to performance problems in your installation? What changes would be most effective in eliminating them?
5. How often does your system experience ENQueue lock-outs? Why? What can you do to prevent them?

Chapter

5

Workloads and Service Levels

MVS workloads consist of five principal categories of address spaces and a sixth category that might be considered a phantom address space:

- Batch jobs—the common workload of earlier operating systems. The normal purpose of a batch job is to perform some data transformation task without human interaction during its execution. In the past, a batch job came into the system on punched cards, and its output left the system on printed paper, on cards, or on both. In today's systems, the input, output, and persistent data are all likely to be on DASD. A job normally does not run continuously but starts and ends, leaving behind some change in the data left in the system, or creating output data that leaves the system.

  Although a job may be divided into individual steps, the job as a whole is usually thought of as a single transaction with a single purpose. The name of the job is specified as the first identifier or token on the "JOB card," the first 80-byte punched-card image that constitutes the job. The remainder of the job control language (JCL) that defines the job may be supplied wholly or partly through one or more *catalogued procedures* contained in libraries known to the job entry subsystem (JES).

- Started tasks—address spaces that provide some kind of

operating system or subsystem function. Started tasks (called "system tasks" in MVS's predecessor systems) normally run continuously. The name derives from the usual means of activating these address spaces. The START operator command invokes a catalogued procedure. The program executed by the procedure remains active in the system until shut down by another operator command or until the system is shut down. The name associated with the started task may be specified in the START command but is typically the name of the catalogued procedure.

- TSO sessions—the system-level means of enabling users at terminals to gain access to most of MVS's functions in the same manner as a batch job, but with an increasing number of TSO-specific productivity tools as well. A TSO session starts at LOGON and ends at LOGOFF, but a session may comprise many unrelated transactions. The name of a TSO session is the *user-id* specified in the LOGON command.

- APPC/MVS transaction processors—a general-purpose type of MVS address space, which can potentially have properties characteristic of each of the three older workload types, although considerations for TSO will in most cases be applicable to APPC/MVS.

- OpenEdition MVS Sessions—another kind of command-driven interactive session, in which MVS presents the user with a UNIX command shell and executes programs in a UNIX environment. As an MVS workload it is much like TSO; however, UNIX program structures are different enough from standard MVS structures to require some special handling.

- Enclaves—this is the phantom work unit, not associated with an address space. The first subsystem to exploit enclaves is DB2's Distributed Data Facility, DDF. Enclaves allow individual transactions to be managed with their own dispatching priorities, classified through the DDF subsystem through workload manager when in goal mode or through the ICS in compatibility mode.

We'll consider each of these work unit types in turn, characterizing the resource usage pattern and performance demands of

each. We might, however, find a different classification useful as well. Another way of regarding MVS workloads is to look at two less precise categories:

- Subsystems—address spaces (or related sets thereof) that serve the needs of other workloads or directly support their end users at terminals. IBM's DB2, IMS, CICS and OpenEdition MVS are such subsystems. VTAM is at once a subsystem on its own and a server to other subsystems. Subsystems are usually started tasks, but they sometimes masquerade as batch jobs.
- Individual address space workloads—ordinary batch jobs, TSO or OpenEdition terminal sessions, or APPC/MVS transactions. In a loose sense, these are the direct end-users of the system, as distinguished from those served indirectly through subsystems.

5.1. Batch

A batch job comes into the system by one of several different means, each of which leads to its placement on a *job queue.* Each job on the job queue is in a *job class* and has a *selection priority.* The selection priority affects a job's placement only within its own class on the queue.

Removing jobs from the queue and placing them into execution is done by the *job entry subsystem* (JES). Two such subsystems exist, JES2 and JES3.[1] They differ in details of job management and in the services they provide, but both perform the same basic functions with respect to job selection and initiation. The JES also manages SYSIN and SYSOUT, standard data streams for job input and output. SYSIN forms part of the job queue along with the statements in Job Control Language (JCL) that define the job. SYSOUT collectively constitutes the output queue. The term *SPOOL*[2] is applied to the collection of JES queues. SYSIN is usually associated with the images of 80-column punched cards, and SYSOUT is usually destined for printers, although other options are available.

The image from which many terms describing MVS job entry

1 There was also a JES1, part of the now-defunct OS/VS1 operating system.

2 SPOOL is an acronym for Simultaneous Peripheral Operations On-Line, a primitive form of multiprogramming found in early operating systems.

and scheduling services are derived is that of a card reader whose input is transcribed to the job queue. There are few if any card readers in today's MVS systems, but a common means of placing jobs on the queue is to direct the card images that constitute a job to a SYSOUT facility called an *internal reader.* A START command to the internal reader moves its contents to the job queue.

5.1.1. Types of batch

In many MVS installations, batch comprises two distinct workloads. The more traditional kind of batch is *production* or *scheduled* batch, standard jobs or series of jobs that perform work essential to the business that the installation serves. The other is *non-production* or *unscheduled* batch. This type of batch represents test runs of production jobs, test versions of subsystems that normally run as started tasks, work submitted by TSO users for almost any conceivable purpose, system maintenance tasks—in short, anything that can run in MVS as a batch job other than production. In some installations, the latter group is further subdivided into classifications like *unscheduled production* and *nonproduction*.

Scheduled production batch may seem boring—the same jobs running the same programs every day, week, or month—but it is often the lifeblood of the enterprise.

The importance of production batch is such that whole *production control* departments exist to watch it, to help it, to restart it after problems occur, and generally to escort its jobs and output safely through the system to the ultimate user. Production batch workloads are usually very predictable in timing, dependencies, and resource requirements. Also, production batch typically runs during the hours when online service either is suspended or has low demand. Thus, system operators and performance management personnel are not usually concerned with acute contention problems or other symptoms of unpredictability in the production batch workload, except when the production workload is the victim of some other workload's bad habits.

In contrast, nonproduction batch often does intrude on other workloads, and its management may influence profoundly the performance of the online subsystems or of production batch. The burden of exercising that management has in the past

fallen heavily on the system's operators, unless the system programmers and service planners designed a resilient and responsive set of system parameters and operational procedures to manage the batch workload.

Starting with MVS Version 5, the intrusive problems associated with unscheduled batch can be overcome with considerably less effort through the use of Workload Manager goal mode. Details may be found in Chapter 7.

5.1.2. Job classes and initiators

Batch jobs enter an MVS system through JES-controlled queue servers called *initiators*. Each initiator occupies and serves a single address space. An initiator is assigned to one or more job classes; the classes are in the order specified in the initiator's JES definition. The dequeuing of jobs is sequential by class order whenever an initiator is idle, and by priority order within a class. For example, if an initiator is "open" for (serving) classes B, J, X, and T in that order, a job in class B with selection priority of 1 (the lowest) will be selected ahead of a job in class J with a priority of 15 (the highest). The priority in this case (a JES parameter) has nothing whatever to do with dispatching priority (an MVS parameter).

Job Classing Schemes

Job classes may be assigned for various reasons. Because classes interact with initiators, the classing scheme and the number of initiators and their class assignments must be considered together.

Serialization Approach. In some instances, production jobs must be run in a strict sequential order. The simplest method for ensuring such processing is to place all such jobs in a single class, to start exactly one initiator for that class, and to make sure that the JES reader processes each sequential stream of jobs in the correct order. A common technique to guarantee that order is to place the JCL for the entire jobstream in a sequential data set or in a member of a partitioned data set, and to start an *internal reader* with the desired set of jobs as its input, thus placing the jobstream on the job queue.

One or more unique job classes may correspond to one-class initiators for managing production batch jobs. Such a rigid structure is not responsive to even the relatively slow rate of change

possible in a real-world workload or to the need for adjusting parameters or dealing with recovery situations. Large data centers with extensive batch production work rely on more flexible control schemes. An extensive example of such a scheme is the job-dependent scheduling feature of JES3, in which jobs are *held* until *released* by the successful completion of a designated predecessor. A supplementary IBM-licensed program is available to provide some of the same capability in a JES2 installation. Independent vendors also offer production batch control packages.

Statically sequenced batch jobstreams are not the last word in dependency scheduling. Many modern production systems do not use canned JCL at all, but include *JCL generators* to build and submit[3] successor jobs when predecessor jobs reach the earliest point at which all dependencies are satisfied. Such systems allow more sensitive dependency control than that available in JCL. To take full advantage of large multiprocessor systems, they may also permit "fork" and support "join" of parallel threads. Since scheduling systems can enforce sequential scheduling and also support multiple parallel jobs, reserved classes and initiators for production batch jobs may not be necessary.

Operator Involvement. Another basis for setting up classes reflects the degree of operator involvement needed for a job in its execution stage. By placing a job in a "hold-for-setup" class, the submitter is relieved of the need for the **TYPRUN=HOLD** or **TYPRUN=JCLHOLD** parameter in JCL. JES holds the job automatically, but most installations still require SETUP or MESSAGE JCL statements for quick and accurate determination of the job's setup needs.

The most usual example of setup is tape retrieval and mounting. As the use of automated tape libraries and very-large-capacity online storage devices grows, the need for this kind of hold processing will diminish. The Storage Management Subsystem ([DF]SMS) of MVS/ESA contributes in two ways to the reduction of tape setup jobs. First, by making better use of different generations of DASD, DFSMS might use lower-performance DASD, such as last year's model, for data sets that formerly might have resided on tape. Second, many tape jobs are *backups* of key DASD or *archives* of disused data sets. SMS, by using

3 **SUBMIT** is a TSO command that uses an always-active internal reader to place a job on the job queue. The term is used in a loose sense to describe any process that places jobs on the input queue.

DASD for near-term archives and making backups and archive jobs more selective and efficient, will minimize the need for tape.

A remaining use for an automatically held job class is to allow submission of jobs that are to be explicitly held, irrespective of resource availability, and run at a later time, perhaps to take advantage of a favorable off-shift billing policy.

Service Expectations. If job classes need not be related to production job status or dependencies or to setup requirements except in unusual cases, what is an appropriate basis for establishing job classes? A common choice is to relate classes to service expectations. In many cases, expected levels or rates of resource consumption are included in the determination of classes. A possible scheme including resource requirements and service expectations, as well as production status and setup needs, might be:

Class	CPU sec.	Priority	Comment, other attributes
A	any	low	default class
B	0–2	high	production class
C	2–20	high	production class
D	20–120	high	production class
E	120–600	high	production class
F	over 600	high	production class
G	0–2	high	systematic class
H	2–20	high	systematic class
I	20–120	normal	systematic class
J	120–600	normal	systematic class
K	over 600	normal	systematic class
L	0–2	high	same as G with setup required
M	2–20	high	same as H with setup required
N	20–120	normal	same as I with setup required
O	120–600	normal	same as J with setup required
P	over 600	normal	same as K with setup required
Q	over 120	normal	long-running nonproduction jobs
R	over 120	normal	same as Q with setup
S	any	normal	guaranteed serialization (single INIT)
T	any	normal	same as S with setup

Class	CPU sec.	Priority	Comment, other attributes
U	any	normal	defer to next shift
V	any	normal	same as U with setup
9	any	high	emergency priority

In this scheme, each job submitted is expected to have the TIME= parameter specified on the JOB statement, except for classes S–V and 9. JES, SMF, and SUBMIT exits[4] check for this entry and substitute the appropriate "systematic" job class, according to the time specified and the presence or absence of the /*SETUP statement or its JES3 equivalent. The absence of the TIME parameter (or a class exempt from TIME checking) leads to a default assignment of class A, a low-priority class. Special classes Q through V may be specified in JCL and cannot be altered by the exits. Class 9 may be specified only by designated "authorized" users whose credentials are checked by exits in conjunction with the installation's resource-access-control system. Finally, the production job classes are selected by exits according to the TIME parameter, in conjunction with distinguished USER names or account numbers.

5.1.3. Batch performance considerations

The performance characteristics of batch jobs are examined here based on the "normal" pattern of batch. That is, a job is *not* a never-ending address space simulating a started task. A normal batch job is a voracious consumer of MVS resources, accepting CPU and I/O service and using real storage until the purpose of the job is done. It may wait for the availability of resources it needs or for services it requires, but there is no limit to the rate at which it can consume service other than that imposed by the hardware and the operating system.

Once a job enters a JES input queue, there are several factors that can delay both its initiation and its execution. These factors include:

- No initiator is open to the job's class.
- The open initiators are serving other classes specified ahead of this job's class.

4 Exits are documented interfaces at which installation-supplied code may be inserted for many purposes, in this case to validate and possibly change the control information for a job.

- Other jobs may have higher selection priorities within the class.
- The job has requested the allocation of devices or data sets that are not available, and the JES is not screening for allocation dependencies. In JES3 systems, allocations may be checked before job initiation. If this option is not selected in JES3, and in all JES2 systems, a job may be started and immediately go into "allocation recovery," with MVS's allocation recovery function prompting the operator to substitute available devices for unavailable ones. This is a decision the operator is unlikely to be prepared for, so the job fails on a time-out or by explicit operator option.
- JES3 is not allowing new jobs, or this job in particular, to start.
- Prior to MVS Version 5, or in compatibility mode, the SRM has reduced the system's multiprogramming level (MPL), this job's domain has had its target MPL reduced, and its swap recommendation value (RV) is among the lowest in the domain. The job is thus unilaterally swapped out, and "SRM Delay (MPL)" may become a significant execution state[5] for the job. The same kind of delay can exist in Workload Manager goal mode, except that the MPL constraints are associated with service class intervals rather than with domains, and Workload Manager adjusts the MPL distribution based on service criteria (goals) rather than on resource-oriented parameters.
- The job's dispatching priority may be low, and the CPU is at or near saturation. Time spent waiting for the CPU may become a significant execution state for the job.
- The job may use resources in such a way as to cause significant operator intervention delays. Placing intermediate data sets on tape rather than on DASD (or, even better, on VIO), for instance, may lead to excessive delay for tape

5 Performance monitors such as Candle Corporation's degradation (bottleneck) analysis (in OMEGAMON II for MVS) show the distribution of execution states, including delay states, for selected workloads. This information is collected by Workload Manager for all workloads starting with MVS Version 5. When a delay state accounts for a large part of a job's duration, a tuning opportunity may exist. Such a condition might also confirm success in resource management. If batch performance is sacrificed (for instance to paging delay) and TSO response time is unaffected, that is very likely a desirable outcome.

mounts. Such data sets are too often used in large linked jobstreams to pass data from job step to job step or from job to job.

- The job uses I/O devices inefficiently. Inappropriate block sizes, multiple data sets on the same device, and fragmented data sets with long SEEKs between extents may all lead to "active I/O" becoming a significant execution state for the job. Applications that perform I/O to repeatedly reread unchanging data, instead of using virtual storage in the address space or in data spaces (in MVS/ESA systems) to retain that data, are simply wasting I/O when a more appropriate resource is readily available.

- The job may use shareable I/O devices that are already in use by other address spaces. I/O queuing may thus become a significant execution state for the job. If the devices are shared with another system, the queuing due to inter-system contention appears as "RESERVE delay" in MVS/370. In an MVS/XA or MVS/ESA system, inter-system device contention appears to be the same as genuine active I/O time, because the queuing and redriving of the I/O is handled by the I/O subsystem. The impact of queuing on the same system can be altered if I/O priority is specified for the job independent of dispatching priority, but the contention is only displaced somewhat to other work, not eliminated.

- The job suffers from significant real storage contention. A sloppy storage reference pattern in conjunction with some level of central (real) storage constraint may cause page-in delay to become a significant execution state for the job.

  Real storage contention (page-in delay) can be a highly variable factor in determining the elapsed time of a job. When the job is run at an otherwise idle time, there may be virtually no page-in delay. On the other hand, if the job is run at a prime-shift peak time, page stealing may repeatedly strip the job's address space of all but the most recently used page frames and lead to very long aggregate page-in delay. The increased paging rate should trigger MPL reduction, leading to unilateral swapping; otherwise the increased activity on the local page data sets leads to slower paging response times and a general slowdown of the system. This picture changes substantially with the

SRM improvements in MVS/ESA SP 4.2. The Working Set Manager component of the SRM makes it less likely that central storage constraint leads to unproductive *thrashing.*[6]

We've seen that the number of initiators, along with the selection and order of the classes they serve, is a more significant delay factor than selection priority. Selection priority is specified by the PRTY parameter on the JOB statement or in the separate PRIORITY statement. It is usually destructive to orderly system management to allow users to specify job selection priority. JES parameters provide for the adjustment of selection priority as jobs age on the selection queue. If control of access to the job classes (as by exits or automation facilities) is sound, such management of selection priority is very effective.

Initiators as a Resource

In JES2 environments, initiators are set up in the JES2 parameters and may be altered by operator action. JES3 may alter initiator settings on its own, according to the policies represented in its parameters as they interact with the workload. We will disregard such dynamic adjustments in discussing initiators. We shall regard initiators as specific (named) resources, one of which must be acquired by each batch job before it may proceed.

In JES2, each initiator may be set to one or more classes, up to the full complement of 36 possible classes.[7] The number of defined initiators (including both "active" and "drained" ones) may range from zero up to the maximum number of address spaces allowed for the current IPL, less the total of started tasks and logged-on TSO users.

Delay Caused by Initiator Settings

Since an initiator is coterminous with an address space, it serves only one job at a time. The interplay between jobs waiting for execution and the availability of initiators is a crucial factor in determining the length of "on-queue" delay for a job. A tactical position that may have had some value in the past may in-

6 Thrashing is a common name for the condition that occurs when useful work is not done while significant resource is used to move pages in and out of central storage.

7 JES3 is different in this respect and is not considered in detail here. A notable difference is multicharacter class names. However, the same queuing considerations apply to both JESs.

crease this delay. That position is the avoidance of "over-initiation," defined simply as admitting more batch jobs to the system than real storage can accommodate.

In the earliest MVS systems, the physical swapping load required to manage a large number of batch jobs through a smaller MPL had a severe impact on the performance of TSO and online subsystems. Since that time, the "extended swap" algorithm along with "contiguous-slot" blocked swap paging was introduced in MVS/SP 1.3.0, significant improvements in paging and swapping were shipped in subsequent MVS/370, MVS/XA, and MVS/ESA releases; and installed and maximum real storage sizes have increased dramatically. Over-initiation is now a disease with few symptoms. In fact, it is more often an indicator of good system health.

The Need for Over-initiation

Under-initiation is a far greater source of inefficiency than over-initiation. Once jobs are in the system, they are managed by the System Resources Manager. The SRM can select jobs to be swapped in or out to ensure maximum throughput, as well as to respond to the installation's priorities. Workload Manager goal mode since Version 5 uses the same mechanisms but makes its decisions primarily on the basis of workload service targets and business priorities. The SRM responds dynamically to load balancing opportunities. For instance, it might swap in a few more batch jobs when 30 TSO users go to a meeting, and swap the jobs (or their successors) back out when the TSO users return. This opportunity is lost if a stingy initiator configuration is selected to avoid "over-initiation" at peak load time. A great deal of system capacity can be wasted in a mixed-workload system when jobs reside out of reach on the job queue rather than close at hand in processor storage or on page data sets.

It is necessary, of course, to ensure that the swapped-out jobs resulting from planned over-initiation are not holding specific resources needed for other work. The pre-initiation allocation screening done by JES3 eliminates this problem to a large extent, and real-time monitoring of ENQueue contention can detect such problems and facilitate corrective or (future) preventive action.

Initiator Classes

Because of the great variability of workloads and installation

service policies, it is impossible to prescribe an "ideal" initiator line-up. Let's study a hypothetical example based on the job classes listed in the table starting on page 108. We'll state some facts, assumptions, observations, and rules and indicate how a selection of 20 initiators might support them. We assume that the relative priority specification applies to both selection priority and dispatching priority.

- Classes B–H, L, M, and 9 are shown as "high" priority.
- Class A is "low" priority.
- All other classes are "normal" priority.
- No initiators for classes U and V may be active on prime shift.
- Classes D–F, I–K, and N–R have substantial resource requirements. Capacity studies at our example installation indicate that only four jobs in these classes should be active at any given time on prime shift. Up to seven jobs may be run off-shift.
- Only one initiator may be active at any time for classes S and T. Batch jobs requiring serialization07IS lization;of batch jobs are in these classes until the jobstreams are brought up to date with dynamic job submission into classes with open initiators.
- Classes B, C, G, H, L, and M are the classes accounting for the bulk of jobs. They should be accommodated with minimum delay. Ten times as many "0–2" jobs run as "2–20," but the "2–20" jobs may be slightly more important than the shorter ones. Average execution times for "2–20" jobs are five times as long as those for "0–2" jobs.

  If the number of short jobs submitted per hour is 100, the number of longer jobs per hour is about one-tenth of that figure, or 10. Execution time for short jobs is 100 relative time units, and for longer jobs, 50 such units. Execution resources are tied up twice as long for short jobs as for longer ones, so we conclude that the shortest jobs need twice as much preference for initiators as the next longer running class.
- Production jobs rank ahead of nonproduction jobs with otherwise identical attributes.
- Setup is irrelevant once a job is released.

In the initiator setup shown below, all class S jobs are selected by initiator 5 before that initiator (and only that initiator) can select a top-priority class T job. Note that no initiators are defined for subsystems such as IMS or CICS; these are best run as started tasks. Additional initiators or classes may be needed for IMS batch message processing jobs (BMPs).

Waiting times in queue by class may be studied with a tool such as the historical component of Candle's OMEGAMON II for MVS, and adjustments to the number of initiators open to a given class and the order of classes served by each initiator can be made if service targets are not consistently met. Other vendors, including BGS Systems and Computer Associates, offer products to model initiator queuing based on SMF data.

Sample Initiator Settings

No.	Classes	Status	Workload Type
1	9DEFINJOQRKPBCGHLMA	Active	Prime shift heavy batch
2	9EFDINJOQRKPBCGHLM	Active	Prime shift heavy batch
3	9FEDINJOQRKPBCGHLM	Active	Prime shift heavy batch
4	9DEFINJOQRKPCBHGML	Active	Prime shift heavy batch
5	9STCBHGML	Active	Serialized classes S, T
6	9BCGHLMA	Active	High-priority batch
7	9BCLMGH	Active	High-priority batch
8	9BCGHLM	Active	High-priority batch
9	9BCLMGH	Active	High-priority batch
10	9BCGHLMA	Active	High-priority batch
11	9BCLMGH	Active	High-priority batch
12	9BCGHLM	Active	High-priority batch
13	9BCLMHG	Active	High-priority batch
14	9CBHGMLA	Active	High-priority batch
15	9CBMLHG	Active	High-priority batch
16	9CBHGML	Active	High-priority batch
17	9CBMLHG	Active	High-priority batch
18	RKPBCGHLMA	Drained	Off-shift deferred
19	9VUEFDINJOQRKPBCGHLMA	Drained	Off-shift deferred
20	9UVFEDINJOQRKPBCGHLMA	Drained	Off-shift deferred

Analysis of a data center's needs or experience may lead to conclusions about initiator lineups that do not resemble the sugges-

tions given above.[8] However, many systems end up with excess effective initiator capacity, so it is difficult to test queueing model predictions or new ideas with rigor. If initiator delay is seen as a bottleneck, a series of carefully tracked incremental changes, with periods of measurement and stabilization between changes, can lead to improvement in a short time.

It's almost impossible to have a problem due to "too many" open initiators unless many jobs all vie for the same data set. Hiperbatch was introduced to deal with this problem. Another facility that may be useful in improving the performance of batch is MVS BatchPipes, a separately priced add-on to MVS/ESA beginning with Version 4. BatchPipes can bring about a dramatic reduction in the elapsed times of linked series of jobs, known as *suites.* Considerations for using BatchPipes include:

- The sequence of jobs in a suite is rigidly structured and the jobs always execute in a predefined order. The sequence may have several parallel paths and may or may not converge.
- Data is passed from earlier jobs to later ones by means of *transient data sets*—data sets that are created in a job and read and deleted by a subsequent job. Data that must be retained may be copied from a transient data set to a permanent one by an auxiliary job.

Given these prerequisites, what BatchPipes does is to allow the transient data sets to be replaced by *pipes*, queues resident in data spaces that allow the writing and the reading to proceed concurrently as jobs that formerly had to execute sequentially now execute in parallel. JCL changes are needed to change the data sets to pipes, and the impact of parallelization on other work and on initiator settings must be analyzed and dealt with.

The potential benefit of BatchPipes is enormous if a problem job suite's elapsed time can be reduced below some critical duration. However, there is a substantial cost in analysis and modification effort to achieve those benefits. The approach taken by

8 Indeed, many would suggest a simpler configuration. This is one of those areas in which historical accretion can lead to unwelcome results. Frequent review of batch queue delays is needed, often leading to adjustments of the initiator line-up. Initiator settings can interact with Workload Manager goal mode; excessive queuing can be seen directly as missed batch turnaround goals. For alternative views, the author recommends studies by H. Pat Artis and newsletter articles by Cheryl Watson.

BatchPipes—virtualizing transient data sets as data queues— may evolve further and eventually yield benefits with less need for human effort.[9]

Batch can be a challenging workload to manage, and what MVS contributes to that management may solve only a small number of problems. Application tuning, even re-engineering, presents rich opportunities to exploit the strengths of MVS and avoid or undo the follies of naive programmers. Given well-written programs in well-structured jobs, proper attention to the business needs of the installation as reflected in the goal mode service policies can create a system environment that deals effectively with just about anything the initiators will allow to run.

5.2. Started Tasks

In the progression from MVT to SVS to MVS/370, and through MVS/XA to MVS/ESA, a dramatic growth area has been the explosion in the number of started tasks. This trend matched the evolution of IBM's high-end operating system from a batch-dominated workload to an online, interactive system vital to the business of the enterprise. It is no accident that a recent step in the progression that started with System/360 and Operating System/360 was named "Enterprise System/390."

The address spaces that run as started tasks in MVS do several different kinds of work:

- provide subsystem services to batch jobs and TSO users. The job entry subsystem (JES2 or JES3) and systems for managing the archiving and recall of data sets, such as IBM's DFSMShsm, are examples.
- manage terminal networks in support of TSO users or transaction-processing subsystems. The VTAM (Virtual Telecommunications Access Method) address space is a current example.
- provide data base subsystem services for appropriately

9 BatchPipes has since been superseded by a more comprehensive offering called SmartBatch. All the benefits of BatchPipes are included, along with extensions to extend the pipes across multiple images in a sysplex and to break down multistep jobs into separate work units so that pipes may be used. The cost of analysis and modification of jobs is thus reduced. Since SmartBatch was new at press time, additional details should be sought from IBM sources including the System/390 World Wide Web site, *www.s390.ibm.com*.

connected address spaces or terminal users of other subsystem address spaces. The several address spaces that provide DB2 services are in this category, as are several data base subsystems offered by non-IBM vendors, such as Computer Associates' IDMS, as well as the data base portion of IBM's IMS/VS.

- support users at terminals with transaction-processing services. IBM's IMS/ESA and CICS/ESA, as well as TP (teleprocessing) monitors from other vendors, are such subsystems. Some of them, such as IMS, have integrated (yet optional) data baseta base facilities, while others have more or less complex file systems as part of the basic offering.

  Complex interactions among such subsystems are not uncommon. In a single system, DB2 might serve IMS-supported and CICS-supported terminals, TSO users through QMF, and batch jobs executing static SQL applications. At the same time, IMS users and CICS users, as well as specially structured batch jobs called Batch Message Processors (BMPs), make use of data bases provided by IMS/ESA's Data Language/I (DL/I). Finally, CICS users may access data bases defined through VSAM data sets.

  Still more complexity is found when products of multiple vendors, as well as home-grown subsystems, are added to the mix.

- provide services usually thought of as part of the operating system. In previous systems and earlier levels of MVS, these services were part of the operating system's nucleus or did not exist. The WLM, GRS, ALLOCAS, CONSOLE, CATALOG, SMS, APPC, ASCH, OMVS, IOSAS, VLF, and LLA address spaces are examples. Many of these are not really started tasks; they are not launched by START commands but rather are address spaces created as part of system initialization.

  In Chapters 1 and 2 we traced the evolution of MVS to its current structure. Cross-memory services made "horizontal splitting" possible, and the exhaustion of available virtual storage through the growth of MVS complexity and services made it necessary. The trend continues through MVS/ESA and OS/390.

- provide special access control and data management services. ACF2 and Top Secret, from Computer Associates, and IBM's RACF and DFHSM are such programs. The storage management subsystem (SMS) of MVS/ESA is embodied in several such address spaces. As system-managed storage evolves, the differing needs of the several parts of SMS will almost certainly lead to its splitting into additional specialized address spaces.
- receive and process measurement data for the system and its workloads. IBM's Resource Measurement Facility (RMF) and Systems Management Facility (SMF) are the best known of such data collectors. RMF data is captured, collected, and developed in the MVS nucleus, then harvested and passed to SMF by the RMF started task. SMF data is created by numerous sources and funneled to buffers, then collected and written out by the SMF started task.
- collect and display (or store) performance, availability, and other data about the system and its workloads. Performance monitors, such as Candle's OMEGAMON II for MVS, Landmark's The Monitor (TMON) for MVS, and IBM's RMF Monitor III, are in this category.

5.2.1. Started tasks—performance considerations

Unlike batch jobs, started tasks are rarely nonstop consumers of resources, nor do they normally "complete" execution unless commanded to do so. Most started tasks spend the bulk of their lifetimes in the WAIT state, depending on the occurrence of some external event (I/O, timer interrupt, or supervisor call) to trigger a burst of activity, and then go back to sleep.

Other started tasks are driven by the demands of MVS workloads through cross-memory services, *via* either SRB scheduling or cross-memory PROGRAM CALLs. After initialization of the started task, subsequent CPU use flows as part of the requester's instruction stream, is at the dispatching priority of the requester, and is charged to the address space requesting service. The started tasks that act as subsystems have minimal intrinsic resource needs but simply reflect the aggregate demand of their end-users.

Many started tasks are key components of the operating system.

As such, they have high dispatching priorities set by MVS, and may have special immunity from page stealing. Unless unusual workload conflicts cause unanticipated problems, these address spaces rarely have, and can rarely cause, performance problems.

Dependencies

Started tasks on which other workloads depend for service must be more favored in MVS than the address spaces that depend on them. Otherwise, the dependent address spaces and their end-users will suffer multiple delays. Consider, for instance, a CICS service that serves 1000 terminals. There might be a CICS terminal-owning region (TOR), two application-owning regions (AORs), and three resource-owning regions (RORs) communicating through cross-memory services to process transactions.

The CICS subsystem depends on other started tasks for necessary services. If the installation is in transition from older data base architecture to a relational data base, both IMS-DB and DB2 (or equivalents from non-IBM vendors) may be present. VTAM is needed to manage the terminal network. IBM's RACF or Computer Associates' CA-ACF2 might be used to provide LOGON security and to control individual users' access to transactions.

When a CICS transaction needs a service provided by one of the non-CICS address spaces, CICS requests it, and the transaction waits until the server signals an end to the *WAIT* by issuing a *POST* SUPERVISOR CALL. Any delay in the server address space, including that caused by the CICS address spaces as they serve other transactions, affects only those transactions waiting for the outside service. On the other hand, if a CICS address space suffers a delay, perhaps to wait for a page fault to be resolved, all transactions will be delayed.

The balancing between internal delays and external delays, between holdups to individual transactions and delays to the whole subsystem, is a complex part of MVS performance management. In this example, as in general, it is essential to have a good understanding of the service targets, the actual degree to which those targets are met, and the nature of the delays causing targets to be missed in order to conduct a sound performance management strategy.

Storage Considerations

There are few MVS systems with absolutely no storage con-

straint. Consequently, the need for and use of virtual and real storage by started tasks must be analyzed and understood along with such need and use in the rest of the system.

To protect an address space from paging delay, the usual method prior to Version 5 was to use *storage isolation*—specifying a target frame count in an address space—to protect it from page stealing. The target may be adjusted based on the paging rate sustained by the address space. Storage isolation is described in greater detail in Appendix B. Its use is less necessary and less effective in systems with expanded storage.

The mechanisms and protections of storage isolation are provided in Workload Manager goal mode without the need or the means for explicit specification of such protection. Instead, a work unit that is suffering from storage delay and which is deemed of sufficient importance to merit assistance will be given a dynamically determined "storage protection" based on its current working set behavior.

Started tasks that provide critical and frequently requested services for the rest of the system must deliver their services in the least time possible. The Global Resources Serialization (GRS) address space is a good example. It runs at high dispatching priority, equal to that of the "master" address space (but ranking after it),[10] and prior to Version 5 it had a default pseudo-specification of storage isolation that protects an increasing number of its frames from page stealing whenever a page fault is encountered.

As we move from the unquestionably "important" started tasks to those whose services have intrinsically longer service times and are requested less frequently, protection from paging delay is less necessary. Even if a CICS terminal-owning region suffers two page faults per transaction, not more than a tenth of a second will be added to each transaction's response time. If the service target is a 2-second response time and it is now 3 seconds, paging in the TOR is unlikely to be the most promising tuning target.

CPU Considerations

Dispatching priority specification mechanisms in compatibility

10 Although the MVS dispatcher works in general on a first-ready, first-served basis for address spaces of equal priority, many MVS service address spaces have a fixed priority of `X'FF'` and a fixed order within that highest priority.

mode are discussed in Chapter 6. For now, let's simply examine the order in which different address spaces should have access to the CPU. There are some simple rules that we can derive from the nature of each type of started task. First, remember that the dispatching priorities of several address spaces are out of reach; they are set by the operating system and can't be changed. In an MVS/ESA SP 4.2 system, these include the MASTER, RASP, GRS, DUMPSRV, CONSOLE, SMF, SMS, APPC, ASCH, IOSAS, XCFAS, and CATALOG address spaces.[11] Because the others *can* be changed, we must understand the range of possible changes and the reasons for making each selection.

If we suppose that there is such an ideal as a "well-ordered CPU," a measure of that order is that high-priority started tasks are essential to other work, that they are needed frequently, that they have short service times, and that they rarely use resources subject to contention, so they rarely encounter delays. As priority for access to the CPU declines, each of these attributes moves toward its opposite extreme. Since we wish to plan such an ordering, we should look for these characteristics and arrange our priorities accordingly.

The top priorities are assigned to those auxiliary address spaces of the operating system for which dispatching priority may be assigned: WLM, PCAUTH, ALLOCAS, VLF, LLA, and TRACE, with SMS added as appropriate. Because most of the activities of these address spaces are invoked through cross-memory PROGRAM CALLs,[12] dispatching priority is of significance only during initialization and recovery, and for those remaining services still invoked through SUPERVISOR CALLs. In Workload Manager goal mode a few of these started tasks are assigned to the built-in service class SYSTEM, which has an automatic assignment of top dispatching priority; the others are assigned to SYSSTC, the next-ranking service class.

The next layer of started tasks includes those required for performance and availability monitoring of the operating system. Those monitors that sample or observe the execution states of other address spaces, particularly in order to measure CPU de-

11 As of Version 5, goals may be set (or dispatching priorities specified in compatibility mode) for all but MASTER. This is not a recommendation that the priorities of these address spaces be changed just because it's possible!

12 Remember that a cross-memory PROGRAM CALL is a simple flow of control in an instruction stream, retaining the dispatching priority of the originating address space.

lay, must do so from the vantage point of higher dispatching priority for accurate results.

Following performance monitors in rank are enabling subsystems. The VTAM address space must be higher in priority than its users, and JES must be "above" address spaces that make use of its services. With JES2, these are all batch jobs and TSO transactions other than the shortest-running ones. JES3 has a more dynamic role to play in TSO systems, so its priority should be above that of all TSO work. Ranking about equal with TSO is the Terminal Control Address Space (TCAS), used only for TSO LOGON. These two layers of started tasks may be assigned (through a special procedure described in Chapter 7) to the second special service class in goal mode, SYSSTC, and are given a dispatching priority just below that of the SYSTEM class.

Finally, having gotten through all of the "overhead" started tasks, we get to those that actually do some work. IMS, CICS, and DB2 subsystems are at this level of priority. Again, those acting as servers to other address spaces must, in general, rank higher in priority than the users of those services. Thus the IMS Control Region is higher in priority than its Message Processing regions, and the DB2MSTR address space is at higher priority than those it serves. This must be so even though most DB2 services are provided through cross-memory calls. In Workload Manager goal mode, the server address spaces may also be assigned to the SYSSTC service class.

At the bottom of the ladder are test counterparts of the production subsystems—started tasks, but not of crucial importance to the main purpose for which the system is installed. Also low in priority are started tasks that are not performance-critical. Miscellaneous data collectors such as the RMF (writer) address space are in this category.

5.3. TSO

TSO (Time Sharing Option) became available in 1969 as an optional part of MVT. The option was a large one; a batch-only operating system had to be extensively modified to accommodate a very different type of workload. Because several TSO users had to be accommodated in a single MVT region, a form of swapping had to be added. Because the data set needs of TSO users could not be predicted in advance, dynamic allocation was a necessity.

An efficient terminal access method was needed, and TCAM was the initial response.

TSO was not a notable success in MVT for a number of reasons:

- Terminals were expensive and slow.
- TSO response time in MVT was intrinsically slow.
- The alterations to MVT in support of TSO were extensive and unreliable, resulting in a system that could no longer be trusted to run the backbone batch workload and could not support a useful number of terminal users.
- IBM offered a superior time-sharing alternative in CP/67, the predecessor of VM/370, with CMS. At the time that TSO was struggling to define itself, CP/CMS was becoming a mature system supporting far more terminals for an equivalent configuration.

From this unpromising beginning, TSO's reputation continued to suffer through later releases of MVT and into the first few releases of MVS. TSO in MVS is no longer an "option," but the name was retained. TSO's problems in early MVS were covered in Chapter 1. Let's look now at the reasons for TSO's good reputation and widespread acceptance and use in more recent versions of MVS:

- TSO was designed into the basic structure of MVS, not grafted on as an afterthought. MVS was also designed for high integrity; more recent versions have been made suitable for continuous operation. Consequently, MVS with TSO is far more reliable than was MVT without TSO.
- VTAM was far more efficient and capable than TCAM at managing large terminal networks.
- Terminals for TSO evolved rapidly from converted typewriters to alphanumeric CRT devices in IBM's 3270 line, with color and graphics being added and prices declining almost as rapidly. A TSO user today might use a personal computer, equipped with any one of several connectivity adapters or a software emulator, as a terminal. Such a terminal might support four simultaneously connected sessions, each emulating a full-function IBM 3179 or 3194, with full native PC functions available as well. Connection

of such an *intelligent workstation* or of an ordinary "dumb terminal" to MVS through a multisession extension to VTAM can provide access to a practically unlimited number of TSO or other sessions.

- More sophisticated interfaces than the primitive TSO line-oriented commands became available, increasing the functional richness and productivity of TSO. The most successful of these is IBM's ISPF (Interactive System Productivity Facility), which has become so standard an interface that IBM has made it a basic element of OS/390.
- Most important, MVS became capable of running TSO well. The introduction of logical swapping was followed by improvements in paging and swapping and the increase in real storage sizes found on MVS systems. These developments eventually overcame TSO's appetite for storage.[13] Because TSO is a basic part of MVS, every performance improvement in MVS has benefited TSO as well.

5.3.1. TSO as a workload

Each interaction of a TSO user with MVS is known as a *transaction.* Transactions are distributed in size, most of them being short, or *trivial.* A trivial transaction typically executes a few thousand to about one hundred thousand instructions. It may call for some small number of I/O operations in addition to the terminal I/O that initiates it, and it may make reference to some 20 to 200 frames of real storage.

A TSO service is usually judged by the speed and consistency of its response time to trivial transactions. In recent years, behavioral research has suggested that response times under one-half second increase the productivity of TSO users, perhaps by changing the pattern of transactions they enter. Other research indicates that consistency of response time is perhaps a more significant factor.

Longer-running TSO transactions may call on complex MVS services or may do far more I/Os than trivial ones. Experienced TSO users tend to realize when a transaction is likely to make heavy resource demands, and they are prepared to wait several seconds for response.

13 Of course, the users learned to throw larger and larger transactions at the system. Working sets of 1500 frames are no longer uncommon.

Another class of TSO users is likely to make more stringent demands and be less tolerant of response delays. These users may be using TSO as a gateway to DB2 through the Query Management Facility (QMF), or using it to access one of several fourth-generation language facilities (4GLs). These users are *not* knowledgable about TSO or MVS, and they are insulated by their application interface from an understanding of how complex their requests are. Such a workload is a great challenge to manage well, often requiring the performance analyst to exercise persuasive skills on the *ad hoc* application programmers while educating them in the realities of MVS performance. Chapter 11 provides some suggestions for dealing with such workloads and those responsible for them.

5.3.2. Transaction profiles

Typical TSO transactions spend most of their time waiting to use the CPU, waiting for page fault resolution, or waiting for the completion of physical swap-in. If a significant portion of transaction time is spent *using* the CPU, waiting for the SRM to allow a swap-in, for I/O completion, or for ENQ conflicts, either the transaction is unusual or the system requires tuning.

A complicating factor in characterizing the TSO workload is a consequence of TSO's generality. In contrast with more rigidly structured terminal-based systems like CICS or the CMS component of VM/ESA, the TSO environment allows virtually any program that can operate in the MVS environment to be invoked in TSO via the CALL command or some variant thereof. If the invoked program simply performs some action and exits, no unusual behavior is evident. However, if the program does what would have been unit-record I/O if it ran in batch, the standard SYSIN DD name is allocated to the terminal, as is SYSPRINT.

Since input records are not ready and waiting for the program, the terminal user needs to enter the input. The "think time" as well as the input entry time appears to MVS as active time rather than idle time. With output to the terminal as SYSPRINT, output will be sent to the terminal until the screen is full, at which time three asterisks ("***") appear at the bottom of the screen and the TSO session is placed in the Terminal Output Wait state. This state, too, is considered as active time (even though the user is not attempting to do anything) and is indis-

tinguishable from a genuine problem-related execution state with the same designation.

Systems in which many TSO users use the CALL command to invoke what are in effect batch programs will appear to have poor and variable trivial response time; in fact, the only problem is corruption of the reported numbers. In Chapters 9 and 10 we'll see how to interpret such numbers and how to correct them for reporting.

5.3.3. Storage considerations

In MVS systems with constrained real storage, paging delay dominates TSO transaction time. A series of actions may be taken to lessen this delay. These will be described in later chapters in the context of the SRM in both compatibility mode and goal mode.

A stable level of TSO storage delay may suddenly increase. A frequent cause of such an occurrence is the installation of a new release of a frequently used TSO facility such as ISPF. In such an installation, the new version is placed in a LINKLIST library, and the modules of the old version in the pageable link pack area (PLPA) are renamed. In MVS/XA systems with the LINKLIST Lookaside Area (LLA) and in MVS/ESA with the extended Library Lookaside Area, I/O increase may be insignificant, but the working set (active frame count) of *each* TSO user might increase by an average of 15–30 frames.

5.3.4. I/O considerations

Because TSO is an online window to all of MVS's capabilities, and because batch jobs can be translated easily and exactly to TSO CLISTs or EXECs, almost any degree of I/O intensity might be encountered in a TSO transaction. It may be presumed, however, that I/O-dominated TSO transactions are atypical. If a group of TSO users do tend to be heavy I/O consumers, it might be appropriate to segregate them, manage them, and measure them apart from others with a differently distributed execution profile. SRM provides such mechanisms and they will be described in later chapters.

An indirect and hard-to-see form of I/O delay can occur in TSO systems, largely as a result of carelessness. Every system programmer should realize that TSO is a program-fetch-intensive environment. If there are five TSO

transactions per second, there are probably 10–20 *LINK*, *LOAD*, *ATTACH*, or *XCTL* SVCs executed by TSO transactions or on behalf of them by TSO application platforms like ISPF. If 95 percent of the TSO fetch activity is resolved from the PLPA, there is no significant TSO I/O impact. However, at many installations TSO LOGON procedures become modified for one reason or another. A common procedure is to use a STEPLIB DD statement in the newly modified LOGON procedure, to keep the new code out of the LINKLIST libraries until it is proven by use. If the STEPLIB is temporarily used for such a purpose, a reasonable tradeoff has occurred. If the STEPLIB is left in effect, the system will suffer.

A STEPLIB is efficient for batch jobs or for a started task subsystem, where all (or at least most) of the program modules to be fetched will be found in the STEPLIB. It may also be efficient for a constrained or closed application system built on TSO. It will be grossly *inefficient*, however, for an unconstrained TSO service, because each fetch operation will cause the entire STEPLIB directory to be searched, to no one's benefit, if the module is in PLPA but not in the STEPLIB.

5.3.5. CPU considerations

CPU delay tends to dominate TSO transaction times in systems without storage constraint. Usually, this situation is innocuous: "The CPU is the last bottleneck." Excessive or inconsistent CPU delay is almost always caused by deviation from the scheme of "well-ordered" dispatching priorities we examined in the discussion of started tasks in this chapter. Again, we examine in Chapters 6 and 7 the methods by which TSO transactions may be placed appropriately in dispatching priority order, and we show how to apply those methods in Chapters 9 and 11.

5.4. APPC/MVS

APPC/MVS is a robust implementation of program-to-program communication in the MVS environment. IBM describes it as "a VTAM application that extends APPC support to the MVS/ESA operating system."[14] APPC, in turn, is a particular implementa-

14 This quotation as well as other material in this section is from IBM's publication *MVS/ESA Planning: APPC Management.*

tion of the SNA (Systems Network Architecture) LU 6.2 protocol.[15]

Pursuing the alphabet soup further, but only to the point at which we can move to a discussion of MVS performance management for the APPC/MVS workload, we note that LU 6.2 is the SNA logical unit meant to handle communications between application programs, whether on the same platform or across a network. An earlier implementation of APPC/VTAM did not support all of the LU 6.2 protocol. APPC/MVS does.

From a programmer's point of view, APPC/MVS provides a set of services, callable from a program, to support communication with other programs using SNA protocols. There is no inherent structure—applications can conform to the client-server model or communicate in a less structured peer-to-peer arrangement. As a system workload to be managed, APPC/MVS includes MVS address spaces, members in SYS1.PARMLIB, operator commands, reporting functions in RMF and SMF, and a need for the full range of workload management attention, including performance management.

Its support structure in MVS is complex and makes use of numerous existing MVS facilities as well as some new ones. Subsystems and applications built on APPC/MVS have no limitations corresponding to those of TSO. A subset of APPC/MVS facilities is portable and equivalent to the CPIC (Common Programming Interface for Communications) definition of SAA. An application written to those interfaces is portable across the SAA platforms, including OS/400 on the IBM AS/400, AIX/6000 on the IBM RS/6000, and OS/2 Presentation Manager or Workplace Shell on personal computers.

The APPC/MVS environment is controlled by two started task address spaces, *APPC* and a transaction scheduler usually named *ASCH*.[16] It also requires an appropriate level of ACF VTAM to accommodate LU definitions as well as to supply communication services to programs on nonlocal nodes. The APPC address space provides session-level management and turns

15 VTAM, SNA, and communication protocols are well beyond the scope of this book and the competence of its author. Several other volumes in this series cover these subjects, and the reader is urged to study these books as well as IBM's documentation before making use of APPC/MVS facilities.

16 We'll assume the ASCH name in the following discussion. APPC/MVS allows the scheduler to have an arbitrary name and to be replaced, provided interface definitions are met.

transaction scheduling over to the scheduler address space. ASCH, in turn, creates and assigns class attributes to the transaction-processing address spaces (TPs). These TP address spaces form the APPC/MVS workload to be managed.

Given the great flexibility inherent in the APPC/MVS structure as well as a general lack of experience with APPC/MVS workloads, it is difficult to describe the resource needs or workload management aspects of this workload. However, the essence of the facility is communication initiated by people. Good response time is therefore important. To a first approximation, APPC/MVS address spaces should be managed like TSO workloads, with each transaction class managed commensurate with business needs. When a client-server structure is established using APPC/MVS facilities, server address spaces should as usual be preferred over clients.

As in any discussion of service, the experience at the installation is the best source of characterization data. Consult RMF or CMF Workload Activity Reports to determine if there is a large number of transaction completions in an interval. If so, a TSO-like model is appropriate. If not, dig deeper into what the APPC transaction processor is doing; if it's a persistent server session it should be treated like any other server address space such as a CICS application-owning region.

There is a pronounced resemblance between CICS as a subsystem and alternative subsystem structures that can be built using APPC/MVS. CICS's vulnerability to paging delays and storage integrity violations is being overcome by internal restructuring as well as by the availability of Subsystem Storage Protection since MVS/ESA SP 4.2.2. However, APPC/MVS is another factor in CICS's future, and inter-system communication based on APPC/MVS may be implemented within CICS. This may be a first step in a restructuring of CICS as a more resilient subsystem. As further evidence of IBM's commitment to APPC/MVS, OS/390 Release 3 includes APPC/MVS extensions for generic LOGON and workload balancing, as well as for distributed transaction processing.

5.5. OpenEdition MVS

As of MVS/ESA SP 4.3, MVS was further extended in an unprecedented direction. Through a series of incremental releases and

using bits and pieces from TSO and APPC/MVS as well as a great deal of new code, IBM erected a complete UNIX structure fully integrated with MVS. As of OS/390 Release 1, OpenEdition MVS complied with all the principal UNIX standards and achieved XPG4 branding; with Release 2 it has gained full UNIX certification. In other words, MVS is not a system with a UNIX interface grafted on. To all intents and purposes, MVS *is* UNIX, while still being all that MVS has been.

OpenEdition MVS (OE) is a subsystem that supports users at terminal sessions, much like TSO. It is also a subsystem in which a process in a terminal session can create an independent additional process in another address space, much as can be done through the base facilities of APPC/MVS. Its sessions can access MVS data sets as well as files in a fully-realized UNIX hierarchical file system (HFS), supported by DFSMS. The vendors of data base subsystems such as Oracle and Sybase, usually thought of as UNIX-centered, have announced versions for the OE environment. In short, OE presents the full range of performance management problems and opportunities as found elsewhere in MVS.

A good starting point for managing OE sessions is to treat the terminal sessions the same as TSO and the subsidiary address spaces ("forked children") like batch. When experience is accumulated, adjustments can be made. In Workload Manager goal mode the performance requirements can be directly stated as goals, so little adjustment should be needed.

5.6. Enclaves

In a subsystem like DB2's Distributed Data Facility (DDF), where work requests of differing business importance can arrive from multiple sources, it's difficult to manage all the work units appropriately given only a single address space's dispatching priority. With all work requests treated equally, the less important ones might be treated too well, or the more urgent transactions could be denied needed resources. Enclaves solve the problem by giving the external work classification system—either the goal mode policy or the ICS in compatibility mode—a chance to "see" each work unit and assign it to a service class or performance group.

5.7. Business Value and Service Level Agreements

All of the workloads that run in MVS use system resources and contend for them. There must be some rational basis for deciding which workload should receive preference in resolving contention. The essential element in making such decisions is business value. A secondary consideration is the intrinsic nature of the workload. If we reflect on the latter, we can see that it still comes down to business value. For instance, if we assert that sub-second TSO response time is good for the productivity of the users, we are saying that such productivity is good for the business and worth the cost of achieving it. However, is it worth more than providing two-second response time for Accounts Receivable users of CICS? Is sub-second TSO response time more important to the business than ensuring that printed output is "in the bin" within 30 minutes of job completion? Clearly, it depends on the business and the set of tasks that constitute the MVS aggregate workload.

The performance need of each workload has two dimensions—a numerical target and a relative business importance. Establishing a service level agreement is a process of establishing each workload's desired performance and then assigning relative business priorities across all of the workloads. Generally, setting the numerical targets tends to be easy while setting the priorities can be a hair-raising exercise in organizational politics and ego clashes. The only advice the author can offer is to develop a good plan to specify when and to whom to escalate unresolved conflicts.

Once the targets (goals) and priorities are worked out, it should be possible to set up MVS so that the required performance levels are achieved. This was a very difficult process prior to the availability of Workload Manager goal mode in MVS/ESA SP 5.1. In goal mode it is straightforward and direct.

5.8. Summary

In this chapter we have examined the workloads of MVS, the ways in which they are brought into execution, and their dependencies and vulnerabilities. We have also taken an overview look at the motivating factors behind performance management activities.

5.9. Chapter Questions

1. Review the set of initiator settings for the batch workload on your system. Is it facilitating efficient operation or limiting it? Has it been adjusted to keep up with hardware upgrades?
2. Is batch (and long-running TSO) managed with an objective of avoiding "over-initiation"? If so, examine the justification for this approach. What measurements would you need to determine whether this strategy is wasting CPU cycles by holding them back from a workload?
3. Chart or list in order the dispatching priorities in your system. How closely do they approach the ideal of the "well-ordered CPU"? What changes would bring the system closer to that condition?
4. How many batch jobs in your system are really started tasks in disguise? Why are they not run as started tasks? If there are functional reasons (not just an arbitrary choice), what needs to be changed to allow them to run as STCs?
5. Is there a rivalry between MVS and UNIX advocates in your installation? Is UNIX as cost-effective as MVS? If not, and it probably won't be, can OpenEdition MVS improve the use of human and computing resources by offering an alternative platform for UNIX users? If there appears to be potential, prepare a plan.

Chapter

6

MVS in SRM Compatibility Mode

> "When *I* use a word," Humpty Dumpty said, in rather a scornful tone, "it means just what I choose it to mean—neither more nor less."
>
> —*Lewis Carroll*

The MVS component that manages the distribution of system resources to workloads is the System Resources Manager, SRM. This chapter introduces SRM's concepts and terminology. We'll cover the basics of time concepts, service units, service rates, swap controls, and working set management. Because the principal focus of this edition is workload manager goal mode, further details may be found in Appendixes A and B.

Of necessity, we must explore the concepts and parameters of the SRM as they were prior to the availability of Workload Manager goal mode. Understanding the complexity of the "old way" is a useful step in exploring the benefits of the current SRM design. Many of the original concepts do survive, although often in altered form.

When we speak of compatibility mode, we use the term in a broad sense to refer to the SRM as it was in MVS/ESA SP 4.3. However, there were specific changes for compatibility mode in Version 5, so we'll point those out explicitly in the discussion.

6.1. SRM—Using Part of the System to Manage Itself

MVS, even in compatibility mode, may be unique among operating systems in the extent to which it uses part of the system's resources to manage the rest of those resources. It almost certainly was one of the first to do so. MVS may also be unique in the subtlety of many of its concepts—and the obscurity of the language that describes them in compatibility mode.

SRM was in the past often described as having two priorities: the first, to allocate resources to workloads in accordance with installation directives; the second, to maximize throughput of the system consistent with the first priority. In that brief description is the key to the core of MVS performance management in compatibility mode. Accurate translation of the installation's priorities to language (in the *IPS*, *OPT*, and *ICS*) that the SRM "understands" ensures that MVS (as directed by SRM) will operate in accordance with those priorities. Failure to do so may lead to unpleasant surprises. In the absence of correct guidance to SRM about workloads, control of the system will be dominated by SRM's second priority, that of maximizing throughput. The throughput priority objective is served more completely by a set of functions that were new in MVS/ESA SP 4.2. These functions, collectively termed the Working Set Manager, aid throughput by optimizing use of central storage, thus minimizing "nonproductive" use of CPU time for initiating and managing paging.

6.1.1. The danger of defaults

In new MVS systems especially, it is all too easy to be overwhelmed by the sheer volume of work needed to get the system functioning at all, and to push aside performance management considerations. This (perhaps unintentional) choice appears sound because everything appears to be working well at first. Until there is contention for one or more resources, MVS will appear to have few performance problems, and all workloads will receive adequate, if sometimes erratic, service. But as soon as a chronic constraint appears, MVS's throughput-oriented default controls take over, often to the detriment of response-critical workloads. The system thus has an intractable dual personality—reasonably well-behaved when there is no constraint, and

to all appearances arbitrarily harming work units when there is a constraint.

IBM recognized this problem in SP 4.2 and later releases by including in the SRM the capability of detecting and responding to the indications of real (central) storage shortage. This capability of the Working Set Manager operates without external controls and avoids the unproductive use of CPU cycles to manage excessive page movement.

Other defaults do survive, with sometimes counterproductive effect. It's not necessary to live with them. With some knowledge of SRM and how it is directed, the system programmer or performance analyst may conduct simple experiments aimed at improving the performance of key workloads.[1] These experiments are easy to assess and safely reversible, yet few bother to try them. The sheer size of MVS intimidates the prospective performance specialist—and so another MVS system continues to run under the control of IBM's defaults.

In many systems that have numerous provisions for adjustment, the defaults might be good enough to accept for a while. However, IBM's defaults for MVS throughput management are in many cases based on arbitrary limits, on extraneous considerations such as the speed of the CPU, or on unconstrained laboratory measurements. They are not related to the management of real workloads, nor are they based on typical configurations.

Although the defaults as of SP 4.2 were in general more appropriate, there is an additional complication of defaults when migrating from an earlier MVS release to MVS/ESA SP 4.2 or later. No one is foolish enough today to expect a system running with out-of-the-box defaults to perform optimally. However, it is reasonable to expect that IBM defaults will not undo carefully implemented performance management actions. If we also believe, based on past experience, that SRM controls are compatible from release to release, we can experience some severe default shocks.

The problem arises from the *diagnose and ignore* phenomenon. When some OPT controls were "retired" in SP 3.1.3, it would have been catastrophic for the entire OPT to be rejected simply

1 Note that performance management deals with workloads. There are some activities, described in Chapter 10, that might properly be described as "system tuning," but performance problems and efforts to conquer them have to do with workloads.

because it contained a now-obsolete parameter. Instead, the old parameters were recognized, syntax-checked, and then ignored or discarded. If an otherwise correct OPT contained, for instance, the obsolete specification

```
PAGERT2=(300,390)
```

it was accepted and the parameter ignored. If the parameter was written incorrectly, as in

```
PAGERT3=(300,390)
```

the syntax error was "diagnosed" (detected) and the OPT rejected.

In SP 4.2, some additional OPT parameters received this treatment, while many parameters in the IPS became *diagnose and ignore* candidates. What was left after all of the ignoring may not have been enough to reflect the installation's resource allocation and workload management policies completely, and this is where the defaults, including some newer ones, come into play. The only way to avoid this trap is to ensure that those policies are translated to the correct current parameters.

6.1.2. Management approach

To "tame" MVS is an exercise in several parts. These are summarized below as if they were sequential steps, but of course they are not. Many overlapping cycles of activity can be involved in establishing initial performance management and in its ongoing adjustment and refinement.

The authoritative reference source for descriptions of SRM parameter sets and their values is the book *MVS/ESA Initialization and Tuning Reference*,[2] published by IBM in individual editions for each major version of MVS, and amended for each new release of the operating system.

The steps in approximate order are:

- Make a preliminary determination of the workload mix [to be] supported by the system, and establish an order of importance corresponding to the business priorities discussed in the preceding chapter. Make sure to distinguish between absolute importance and relative importance. It's very rare

2 The short title *Initialization and Tuning* will be used in subsequent mentions of this publication—or publications. As of SP 4.2, there are two such books, an *Initialization and Tuning Guide* to explain concepts and an *Initialization and Tuning Reference* showing all the parameters and operands.

that absolute preference for one workload over another is necessary.

Distinguish if appropriate between preference in times of normal operation and preference in times of severe resource constraint. For instance if the aggregate workload runs split between two processor complexes and one fails, key elements of the combined workload may have to run on the remaining system. In such a constrained situation, considerations of end-user productivity may have to be sacrificed to those of keeping the business running.

- Create an initial service level target for each major workload on the system, based on prior experience or the stated requirements or expectations of the system's users. Gary King's "Workload Characterization"[3] methodology is a good starting point for predicting the performance characteristics and resource needs of each type of workload.

- Ensure that the system's hardware configuration is approximately adequate for the projected full workload. Again, King's "Workload Characterization" provides good guidance. King's "Processor Storage Estimation" is a more detailed look at the storage requirements by workload type.

- Reconcile the hardware configuration with the service expectations, and make adjustments as needed.

- Create a set of global SRM controls in an IEAOPT*xx*[4] member of SYS1.PARMLIB so that SRM's ability to vary the multiprogramming level (MPL) of the system will be used effectively to control paging delay of critical workloads by in turn controlling the page fault rate. This important control, as well as others in the "OPT" member, are described in Appendix B.

3 Gary King of IBM's Poughkeepsie Laboratory has published and presented under this title and on the related topic of Processor Storage Estimation in various forms and venues, notably SHARE and CMG. The Processor Storage Estimation presentation was given at CMG '91 and SHARE 78. The Proceedings for each meeting contain a full paper with step-by-step methodology.

4 The name is not an acronym but an abbreviation, and derives from an early version of MVS in which the predecessor of today's SRM was known as the "optimizer"—hence the name "OPT" for its original parameter set. The "*xx*" suffix is the name by which a particular parameter set is designated. The default designation is "00."

In systems with expanded storage, a parallel step corresponding to setting up MPL control is to prioritize the need of each workload for expanded storage and to create appropriate OPT entries and matching IPS parameters to implement a management strategy.

With the ability as of SP 4.2 to control access to expanded storage on an individual workload basis, both MPL control and expanded storage management must be evaluated and implemented for each workload segment.

- Divide the overall workload of the system into distinct *performance groups*; first on the basis of similar types (batch, TSO, subsystem) of address spaces; and second, according to priorities established for the various workload constituents. Establish how TSO, APPC/MVS, OpenEdition, and batch may be expected to break down into short, medium, and long transactions. For each division of these swappable workloads and for each of the nonswappable address spaces, lay out the order of dispatching priorities from the top down, according to the principle of the "well-ordered CPU" discussed in the preceding chapter.

  In an MVS/ESA system at the SP 4.2 level or later, each distinct kind of workload should also be in its own domain, associated with the performance group period definition. In prior levels of MVS, this association did not have to be one-for-one, especially for nonswappable work. This is no longer true.

  Translating these general judgments into the precise language needed for an Installation Performance Specification (IPS) is discussed in Appendixes A and B.

- Once the workloads are associated with performance groups, make sure that the system enforces that association. Do this by creating an Installation Control Specification (ICS), another member (IEAICS*xx*) of SYS1.PARMLIB. The ICS is described in Appendix B.

- Create an overall system parameter set (IEASYS*xx*)[5] denoting the new ICS, IPS, and OPT, along with all the other parameters required to bring up an MVS system. This in-

5 The default name for this parameter set is IEASYS00, often pronounced as "eye-easy-Sue."

nocuous-sounding step is absolutely essential as of MVS Version 5. Prior to Version 5, omitting the ICS, IPS, and OPT parameters from IEASYS00 simply resulted in selection of the default -00 version of each parameter set. In Version 5, omitting the entries from IEASYS00 results in the system initializing in Workload Manager goal mode—even if you're not ready for goal mode.

- Bring the system up with the new parameters. This may not be a simple matter if it is attempted all at once. Since the OPT, IPS, and ICS may be set and reset at will, always keep a proven older version of each of these SYS1.PARMLIB members as the –00 version, making and proving all changes in a member with a different suffix. When all members are functioning as planned, suitable renaming can be done. Conflict will be minimized if changes are made in the order OPT first, then IPS, then ICS.

- Measure the performance of the system and its workloads with tools that can measure service attainment against targets and identify the nature of bottlenecks if they are present. IBM's RMF Monitor I,[6] by itself or as interpreted by IBM's Service Level Reporter (SLR) and its successor Enterprise Performance Data Manager/MVS (EPDM), Computer Associates' MICS, or Merrill's MXG, is one such source; Candle's OMEGAMON II for MVS is another; and other vendors offer similar monitors and reporting programs as well.

  SMF is a general data repository, receiving data about CICS transaction response times, and collecting batch job data. Subsystems like SLR or EPDM, MICS, and MXG format, analyze, interpret, and present SMF-based information.

- Continue the cycle of refining and adjusting the system parameters until all service targets are comfortably and reliably attained. If doing so proves impossible, targets may need to be changed, or configuration changes (such as readjusting storage allocation across LPARs) may be necessary.

- Continue to revise and adjust the service targets until the service delivered by the data center matches what the

6 Boole & Babbage's CMF is an alternative data collector to RMF Monitor I.

management of the enterprise (your "customer") wants, needs, and is willing to pay for.

- Maintain communication with your customers to ensure mutual understanding of current and future needs and capabilities, and monitor the system continually to avoid surprises for yourself or for your customers.

This appears to be, and is, a long and ugly process. We'll wander around for a while in the terminology of SRM and then, at the end of the chapter, explain what's wrong with this procedure and the model that underlies it.

6.2. Introduction to SRM Concepts

The SRM is complex and often appears to be in conflict with itself. It evolved to its final form from the first days of MVS with continuous additions and few if any deletions. Exceptions to this process of accretion occurred with MVS/SP 3.1.3 and again on a larger scale with MVS/ESA SP 4.2. Both of these eliminated controls, and SP 4.2 introduced new controls along with making some long-overdue sweeping deletions. To study the SRM, we need to break down its functions and perhaps oversimplify.

The SRM controls several aspects of MVS, including:

- assigning and dynamically altering dispatching priorities
- controlling page stealing, including protecting pages of favored address spaces from page stealing
- determining which page frames are eligible to be moved to expanded storage
- keeping track of how the CPU resource is used, detecting wasteful use for excessive page movement and acting to optimize the use of central storage
- responding when TSO transactions end, by determining whether the address space will be swapped out to auxiliary storage or retained in processor storage
- changing the multiprogramming level by swapping address spaces into and out of real (central) storage
- determining which tape drive is picked when an allocation request does not specify a device number

- determining which DASD is selected to satisfy nonspecific allocation requests

These are all essentially simple actions; the complexity is in how SRM gathers the information on which its decisions are based, and the specification of the values that guide those decisions. All of these topics will be covered except the last, which is somewhat beyond the book's intended scope.

6.3. SRM Terms and Concepts

To explore SRM and break down its complexity, we need first to acquire its vocabulary. The documentation of SRM uses what appear to be common words to describe concepts, actions, and parameters. Some of these words have uncommon meanings or interpretations in the context of SRM. In this section, we'll explore SRM terms and highlight special meanings as appropriate.

6.3.1. Time in SRM

Various measures of time are used in SRM. Most of these measures are various intervals of normal clock time, but there are others whose basic unit is adjusted to represent ticks of a clock that runs faster as CPU speed increases. Some times encountered in performance management are:

- SRM measurement sampling interval (RM1)—an adjusted time based on the *SRM second*, described below
- time unit—a subdivision of the SRM second
- SRM measurement summarization interval (RM2)—a clock time interval, but not the same in all systems. It varies in duration according to CPU power, and is less than 2 seconds for most MVS/ESA-capable CPUs.
- transaction "response" (completion) time—a clock time (real-time) interval from the time SRM recognizes the start of a transaction to the time recognized as the end of the transaction. [7] Note that transaction completion times are *not* necessarily response times. If transaction completion is signaled to SRM after the end of each terminal interaction,

7 SRM is notified of transaction state changes by means of SYSEVENTs. SYSEVENT SUPERVISOR CALLs are issued by various MVS routines. SYSEVENTs are described in the IBM publication *Component Diagnosis and Logic: System Resource Manager for MVS/ESA* (or its current successor).

the difference is negligible. In TSO systems with (old) non-TSO programs invoked via the CALL command, the difference can be great.

- resident-time—the time an address space spends swapped in during the current swapped-in interval, executing, waiting (involuntarily) for a resource, or in a voluntary wait not designated as "long" (short wait).
- out-time—the time an address space spends swapped out in the current swapped-out interval while the current transaction is ready to run (also known as "out and ready").
- long-wait-time—time spent swapped out in the current swapped-out interval while not ready to run ("out and waiting").

  An address space is swapped out in this state when a WAIT macro with the LONG operand is specified, an STIMER of more than two seconds is issued, or a "detected [long] wait" occurs. The SRM detects an "unplanned" long wait when a voluntary short wait exceeds two seconds or eight SRM seconds, whichever is less.
- active-time—sum of all resident-time and out-time for a transaction.
- transaction-elapsed-time—sum of all active-time and long-wait-time for a transaction.

SRM Seconds

It is a reasonable goal that the impact of SRM overhead be approximately equal across different processor models. Also, the timing of SRM decisions should in most cases be based on the ticks of the CPU's cycle time, rather than on those of a wall clock. Accordingly, most internal SRM timed events occur on the basis of the SRM second. The SRM second was equal to a second of common wall clock time when MVS ran on the IBM System/370 Model 155-II, a long time ago.

For other CPU models, the size of the SRM second is adjusted to represent the time needed to execute approximately the same number of nominal instructions as ran on the 155-II in one second. For IBM systems capable of running MVS/ESA, the SRM

second ranges from about one-third of a second down to less than a seventieth of a second. Consequently, SRM collects the resource information on which its decisions will be based at an interval ranging from less than one to about 70 times per second as CPU speed increases. Note that the duration of an SRM second does not depend on the number of processors in a multiengine complex or on the allocation of CPU resources across logical partitions.

A table in the *Initialization and Tuning Guide* provides SRM second values for IBM processors. Other vendors supply routine MVS system modifications to enter the values for their processors.

Time Units

On slower systems, in which the SRM second may be a time interval perceptible by a person, it might be inappropriate to alter dispatching priorities at intervals as large as one or more SRM seconds. SRM provides a way to subdivide the SRM second into up to ten *time units*. Using this capability, time slicing can operate at an interval smaller than that which a human can perceive—at the cost of additional overhead. The processor speeds of current systems make use of this parameter almost unnecessary.

6.3.2. Service and service units

When a transaction runs in MVS, it uses the basic resources of the system: instructions in the CPU, page frames of central storage, and I/O operations. Because systems differ widely in their ability to provide these resources to workloads, standardized measures are defined to facilitate management of both system resources and workloads.

The standardized measures are *service units*, defined separately for CPU, central storage, and I/O activity including JES SPOOL transfers. The CPU component is further divided into two parts, for TCB time and SRB time.

CPU and SRB Service Units

CPU service is accumulated in two different modes, each representing a separate count of service units. The unqualified "CPU" service is that received in TCB mode. TCB is a Task Control Block, the usual unit of dispatchable CPU work. Other service is received indirectly in SRB mode. SRB is a Service Request

Block, the means by which authorized programs can request MVS services.

Global SRBs are dispatched ahead of local SRBs and all TCBs; local SRBs are dispatched ahead of TCBs in each address space.

This treatment has changed as of MVS/ESA SP 5.2.0 with the introduction of dispatching enclaves, preemptible SRBs, and client SRBs. The new types of SRBs accumulate processor service as "CPU," not as "SRB" time. The time is charged to the address spaces on whose behalf the SRB runs: the home address space for preemptible SRBs, the client address space for client SRBs, and the owner's address space (*e.g.,* DDF) for enclaves.[8] On the other hand, nonpreemptible SRBs are charged to the requesting address space.

A CPU (or SRB) service unit is an interval of CPU time differing by processor architecture and engineering generation, by basic CPU model, by number of processors,[9] and by system environment. In early MVS systems, a service unit was defined as 10,000 instructions. The measure today is in terms of a specified number of service units per second of execution time, or its reciprocal. A second yields from about 80 to over 3000 service units, depending on CPU model. A service unit, then, ranges between about 300 and 12,000 microseconds of CPU service.

I/O Service Units

There are two alternative measures for I/O (IOC) service units. In all MVS/370 systems, and by default in XA or ESA systems, each *EXCP*[10] is counted as an I/O service unit. (In XA and ESA, EXCPs are replaced by a count of I/O blocks transferred, a more

8 This accumulation is as described in SRM's service measurements. For reporting in SMF records, some of these "charges" may go elsewhere depending on the subsystem and the record type.

9 In contrast with the SRM second, the number of CPU service units per second of CPU time is devalued for multiengine configurations, representing the "MP penalty." The loss of productivity from uniprocessor to dyadic, or to other MP configurations, may be calculated by comparing values in the table of CPU service units per CPU second, to be found in *Initialization and Tuning* [*Guide*].

10 *EXCP* stands for EXecute Channel Program, a basic unit of work presented to the MVS I/O Supervisor (IOS), and is also the name of a low-level I/O macro-instruction supplied with MVS. EXCP is used as a last resort when standard access methods lack necessary function or ultimate performance. EXCP has often been thought of as an interface upon which access methods are built. This was never strictly true, and it became clearly false when VSAM and later IMS (OSAM), the JESs, and DB2 supplied their own "I/O drivers."

consistent and inclusive measure.) The second option in XA and ESA, specified by the IOSRVC parameter in the IPS, is to count connect-time, divided by a constant of 0.0083 to make the count roughly equal to that of blocks for half-track records on 3380s. Regardless of which IOSRVC option is chosen, the number of JES SPOOL blocks transferred is added to the IOC total for the JES address space.

Counting EXCPs or blocks gives fewer service units for efficient programs with large I/O block sizes. Even though data bases have become commonplace, much MVS I/O is sequential, and many block sizes are far below optimum. On the other hand, counting connect-time ignores the added disruption of inefficient I/O (other than properly penalizing long SEARCHes), and ignores VIO transfers as well as the CPU overhead associated with each I/O operation. It also devalues I/Os serviced as hits by cache controllers or nonsynchronous DASD subsystems. (If all controllers are cache controllers, there is an appropriate penalty for misses.)

Counting connect time might be the right choice in large multi-engine systems with abundant real storage, but continuity with measurements from MVS/370 systems is lost, if that consideration is still valid. On the other hand, Workload Manager goal mode uses only COUNT. Hence the selection is not obvious.[11]

Main Storage Occupancy (MSO) Service Units

MSO service units are accumulated for central storage held while CPU cycles are being used. The basic unit of measure is a page frame held for one CPU (TCB) service unit. To make MSO roughly commensurate with CPU service units, the raw number is divided by 50 to yield MSO service units. Storage used by cross-memory reference is charged to the target address space (but note that such address spaces do not accumulate CPU service units when accessed in cross-memory mode). MSO service does not reflect expanded storage occupancy.

MVS does not take page frames away from an address space unless asked (as with a PAGE OUT SVC) or until the available frame queue is depleted. Therefore MSO does not indicate *demand* for or

11 The inconclusiveness of the analysis leads the author to suggest another possibility, which should be feasible: why not count both, with different coefficients?

use of storage service. For the other classes of service, the number of service units represents actual demand for and use of service. For MSO, this is not so, and the absence of expanded storage service or SRB time in the calculation makes MSO even less useful and relevant.

Let's examine the contrary nature of MSO service units as a measure of service demand. Consider a batch job step that executes a single program with a total of 3 megabytes (3,145,728 bytes) of virtual storage used from beginning to end. At any time during execution, the program has a true working set of 200K (204,800) bytes, but the entire 3 megabytes is obtained (*via GETMAIN*) and initialized at the beginning of execution and is not freed until the step ends. The job requires 768 virtual pages, but its 200K true working set needs only 50 central storage frames at a time. Suppose also that the job takes 10 seconds of CPU (TCB) time on an IBM ES/9000 9121-621. Thus the 10 seconds of TCB time is about 14,600 CPU service units.

If the job is run in the dead of night, it will acquire 768 frames and keep them until completion. It then uses 768 x 14,600 / 50 = 224,256 MSO service units. The same job run on a fully loaded system at 10:30 a.m. will tend to have its excess frames stolen and will run at close to its true working set. If the average central storage it holds is a little more than its rock-bottom working set, say 250K bytes or 62.5 page frames, its MSO service will then be 62.5 x 14,600 / 50 = 18,250 service units. Note that the CPU service units are indeed commensurate with the MSO service units, but only in the case of storage constraint.

One pursuing a rational performance management approach would tend to run the job at night when resources are more freely available, but someone looking at the reported service units might conclude that less resource is consumed by running the job during the day. The measurements may lead to a counterproductive response.

It should now be clear that using MSO to make performance management decisions can lead to inappropriate results. The use of MSO is even more disastrous if reported service consumption is used for cost recovery (chargeback). In this case, those submitting deferrable jobs will be discouraged from following good habits by simply comparing bills for different runs of similar or identical jobs at different times.

Service Definition Coefficients (SDCs)

We've seen that the raw MSO service units should be suppressed or at least deeply discounted. In systems with constrained I/O resources, we might want I/O activity to count more heavily in performance management decisions. Such flexibility is available. To move from raw service units to the *service* that the SRM uses in various algorithms involves an additional scale factor for each category of service. These factors are specified in the IPS[12] as *service definition coefficients*, and each ranges from 0.0 to 99.9 (or 0.0000 to 99.9999 for MSO only as of SP 3.1.3). Thus *service* is the sum of the four service unit calculations, with each one first being multiplied by its respective service definition coefficient. This scaled and weighted measure of service is also denominated in service units, and we will mean this kind of service unit in the rest of the SRM discussion, unless otherwise noted. Note that the very same structure continues to be used in Workload Manager goal mode, both for reporting and for pacing transactions through service class periods. The raw CPU service number (including SRB time), expressed as a rate, SUs per second, is also used in goal mode for defining and managing resource groups.

It can be tempting for those in charge of resource accounting and chargeback to use weighted service units as a measure of resource consumption. As results are evaluated, they might wish to alter the SDCs in pursuit of consistent billing. However, some users may still continue to receive bills or usage reports that they dispute on the basis of inconsistent charges, while the SRM is denied the use of stable SDCs that allow it to control the system properly.

Altering the SDCs for purposes unrelated to performance management is dangerous and unnecessary. It's unnecessary because the raw components of service are reported in SMF records for each type of workload, or may be calculated by dividing each component by its SDC at the time of data collection. This may then be multiplied by any appropriate factor for billing purposes. The SDCs are readily available in RMF or CMF workload records in SMF.

We might even conclude that SDC alteration (for cost recovery

12 The SDCs are present in Workload Manager goal mode as well. They are in the Service Definition, as described in Chapter 7.

reasons) is foolish, since the degree to which a resource is constrained (the basis for performance management decisions) has little correlation with its marginal cost.

On the other hand, altering SDCs to compensate for resource constraint in performance management can be prudent and useful. However, it should be done with the greatest caution and as infrequently as possible, since small SDC changes can produce significant changes in system behavior. It may be very desirable to shut down the volatile and contrary effect of MSO service units by setting a very small or zero MSO SDC.[13] Even such a beneficial change should be done once, followed by a relatively long period of measurement and evaluation before another change is contemplated.

6.3.3. Service rates

Several SRM algorithms depend on the *service rate* received by a workload. We define service rate as the service received by a workload in a particular time interval, divided by the length of the time interval. The unit of service rate is service units per second. Let's examine some typical examples to understand the rationale for the various ways in which service rates are calculated.

- A batch job has received some service and is now swapped out by the SRM to allow another job to be swapped in. As the job "ages" in the swapped-out state, we would like it to become increasingly eligible for swap-in.
- A TSO transaction is swapped out because it has entered a WAIT state while an archived data set is being retrieved. When the WAIT is satisfied, we would like the session to be just as eligible for swap-in as it was at swap-out time. As it stays swapped out but ready for swap-in, we would like its swap-in eligibility to increase.
- When an address space is swapped in, it may accept or *absorb* service at a high rate. Using only this high *absorption rate* to determine swap eligibility might result in unstable

13 The difference between zero and the minimum MSO value is thought by some to be significant—if it's necessary to retrieve the "frames-held" information given only the MSO service units number. Doing so is impossible if the coefficient is zero. Significance may be lost with a very small value. However, since raw page-seconds are now recorded in the SMF Type 30 record, the charade is unnecessary. **Recommendation**: set MSO to zero.

sequences of swapping in and out. We wish to take account of the swapped-out time, during which no service was received, as well.

Given these goals, we define various types of service rates as follows:

- The count of service units is reset to zero and service starts accumulating at each swap-in time. (Overall service is accumulated for a job or TSO transaction in other accumulators. We are concerned here only with service *rate* as used by SRM.) This count of service units is the numerator or dividend for the service rate calculation.
- For a swapped-in address space, the denominator or divisor is the sum of the most recent out-time and the current resident-time, or the time spent in the current transaction, whichever is less. The service rate will thus start at zero (because the service unit count is reset) and increase until the out-time becomes insignificant compared to the resident-time. Defining *absorption rate* as a service rate with the out-time disregarded, we see that the service rate approaches the absorption rate (if constant) for a swapped-in interval, over time.
- For a swapped-out address space, the denominator or divisor is the sum of the just-completed resident interval and the current out interval. Service rate will start at the value just prior to swap-out and decline linearly over time. Note that long-wait-time is not used in the service rate calculation.

6.3.4. The "workload" scale (obsolete)

The word "workload" had a peculiar and unintuitive meaning in MVS prior to SP 4.2. Fortunately that term was restored to its plain meaning in 4.2, so the Workload Manager didn't have to have some other name. The details of the old "workload" scale and everything that flowed from it are preserved for antiquarians in prior editions of this book.

6.4. New Solutions in MVS/ESA SP 4.2

Concepts that are difficult to explain and parameters and controls that are difficult to use often turn out to be essentially flawed. IBM has recognized this aspect of some of the controls

that drove the SRM, and eliminated some of the most difficult, replacing them with simpler controls and a more sophisticated mechanism with no controls at all.

The **WKL** scale, along with everything based on it, is gone. This includes **OBJ**s, **DOBJ**s, **AOBJ**s, and the load balancers. Let's look at the replacement controls and how they correspond to the former ones.

DSRV and ASRV

Instead of allowing the service rates received by the address spaces in a domain to drive the contention index through the intermediary of an **OBJ**, two new, more direct, controls were introduced, replacing **DOBJ** and **AOBJ**:

- **DSRV**—specified as high and low service rates for the aggregate of all address spaces in the domain, as in

  ```
  DSRV=(5000,50000)
  ```

  When the aggregate service rate is within the DSRV limits, the contention index for the domain varies between 1 and 100 according to the formula:

  $$\text{Contention Index} = 100 \times \frac{\text{DSRV}_{high} - \text{Service Rate}}{\text{DSRV}_{high} - \text{DSRV}_{low}}$$

  Service rates below the minimum result in higher contention indices, up to a maximum of exactly 655.35 at zero service. Service rates above the maximum lead to contention indices between zero and one.

- **ASRV**—specified the same way as **DSRV**, except that the service rate is the average across ready address spaces in the domain.

Load Balancing and the Working Set Manager

The purposes of CPU load balancing and I/O load balancing were already addressed by other existing controls. Storage load balancing is replaced by something new.

SP 4.2 includes a set of functions called the Working Set Manager (WSM). WSM is an elegant and comprehensive solution to the problem of detecting the indications of central storage constraint and eliminating its harmful consequences. Because WSM optimizes the content of central storage and thus decreases wasteful use of CPU cycles, CPU loading might sometimes appear to decline, but the amount of productive work the CPU is doing should not decrease.

WSM detects excessive paging or page movement by distinguishing between *productive* CPU time (usually captured as SRB or TCB time in address spaces) and *unproductive* time (notably SRB time in MASTER). Unproductive time is spent managing paging and includes the CPU cycles for inbound page movement from expanded storage.WSM can cause new types of swap-outs when a high level of unproductive time is being accumulated. WSM also invokes preferential page stealing from heavy users of central storage. Swapping and preferential page stealing are used as alternatives in some cases to allow greater flexibility in responding to central storage constraint.

Associated with WSM is block paging, used increasingly in recent MVS/ESA releases to avoid the repeated disruption of servicing multiple page faults in groups of pages stolen at identical UICs.

WSM recognizes several categories of address spaces. It does not take any action with respect to nonswappable address spaces or those that are storage-isolated, because preferential page stealing and swapping are its methods used to achieve optimal storage use. Of the swappable address spaces that are not storage-isolated, some may be *monitored*, of which a subset may be *managed*. The rest are neither monitored nor managed.

When the ratio of unproductive CPU time to total time for the system as a whole is below some threshold (in the neighborhood of 5 percent), WSM is idle, except for tracking the CPU time ratio. When the system-wide ratio exceeds the threshold, WSM selects address spaces for monitoring, based on their paging and page movement rates. For monitored address spaces, WSM calculates a recommendation value based on a measure of the "cost" (in system resources) of swap-in that can override the recommendation value used for deciding whether a swap-in is due. To prevent undue delays caused by continual bypassing of address spaces, TSO/E users will be swapped in, regardless of the cost, after 30 seconds of out-time, and other address spaces after 10 minutes. Also, implicit block paging from expanded storage is turned on for monitored address spaces, and WSM collects additional data relating to block paging success. If criteria of success are not achieved, implicit block paging may be discontinued.[14]

14 Note that implicit block paging *out* and implicit block paging *in* from auxiliary storage are always active for all workloads. The cost of such block paging, in which a page fault causes an entire group of related pages to be brought in at once, is relatively low for auxiliary storage compared with the potential benefit. The cost/benefit tradeoff for block paging in from expanded storage is less

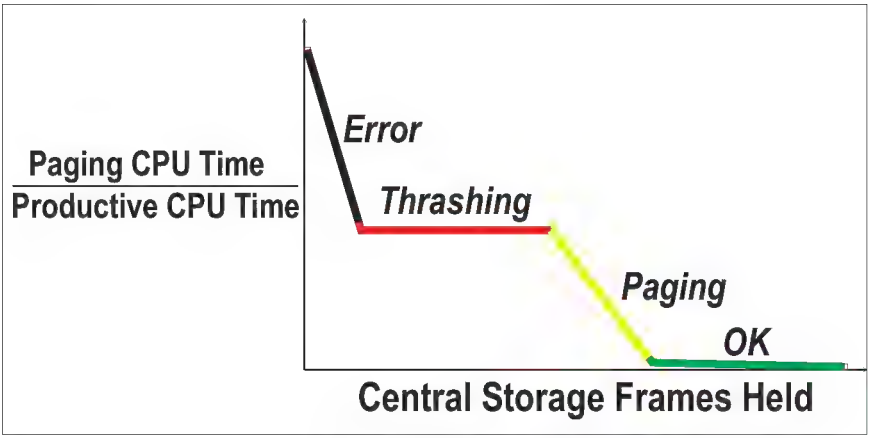


Figure 6-1. Regions of Paging Behavior.

As a monitored address space receives service, WSM tracks its paging against its frame count, dividing the frame count range into four regions as shown in Figure 6-1.

- OK region—above some frame count, adding frames does not materially reduce paging rate.
- Paging region—above a lower frame count but below the OK threshold, adding frames reduces paging rate linearly.
- Thrashing region—above a still lower frame count but below the paging threshold, adding frames does not materially reduce a high paging or page movement rate.
- Error region—the range of frame counts below the thrashing threshold.

When monitoring is in effect and the level of system resources devoted to paging continues to be excessive, a monitored address space will become managed. WSM chooses the monitored address with the highest ratio of paging CPU to total CPU as a function of allocated central storage frames. What WSM does in such a case is to impose an *implicit dynamic central-storage-isolation maximum working set*. The goal of management is to ensure that managed address spaces do not waste their time paging. If possible, enough frames are allocated to keep managed address spaces in the paging range, taking frames from those in

favorable, so that form of block paging is left active only when successful in avoiding page faults.

the OK range, above the dynamic target. If CPU resources are limited as well, a managed address space will be swapped out to improve central storage utilization.

These WSM actions replace and go well beyond storage load balancing, with an automatic mechanism requiring no external specification and basing its actions on actual address space behavior at times of resource constraint. The same mechanisms are used in Workload Manager goal mode although the objective of optimizing storage utilization is subordinate to that of managing workload performance to goals.

Swap Recommendation Values

The idea of swap recommendation value survives to and beyond MVS/ESA SP 4.2. Because there are no **OBJ**s, there is no way to make address spaces in a domain have different RVs given equal service rates. Instead, SRM once again makes workload management decisions influenced by system resource use. For instance, SRM calculates a "cost" associated with swapping each address space in or out. We may simplify by regarding the cost as some function of the address space's swap group size—the number of pages to be swapped in or out.

The swap RV is a number between -999 and +998. Swapped-in address spaces have RVs ranging from 100 (just after swap-in) to 0 (when just enough service has been received to equal the cost of a swap-out) and down toward -999 as additional service is received. An address space that is swapped out and ready to be swapped back in starts with an RV of 0, which increases with time. When the RV passes 100, the address space is eligible for swap-in. A low value (less than 10 and down to 0.1) for an OPT parameter, **SWAPRSF**, increases the rate at which RV for a swapped-out address space increases. A higher value of **SWAPRSF**, to a maximum of 100, retards the increase of RV and discourages exchange swapping. (Exchange swapping is discussed more fully in Appendix B.)

6.5. Changes in Version 5 Compatibility Mode

IBM cleaned out some dangerous deadwood and made a few improvements in compatibility mode for MVS Version 5:

- The APGRNG parameter, used to set the range of dispatching priorities to be managed by SRM, allowed an incom-

plete range to be specified. This had the potential of allowing high priorities to be given away and left uncontrolled. The parameter is removed in Version 5 compatibility mode, and all dispatching priorities are owned by SRM. A consequence is that an installation with the maximum value of APGRNG set to less than 15 is exposed to poor performance for any jobs which had specified an unmanaged high priority.

- The default special storage isolation for GRS is removed
- Multiple performance group periods are supported for non-swappable work.

Other changes that touch on matters not discussed in this overview are detailed in Appendix B.

6.6. The Basic Flaw in the SRM Model

The numerous steps in the process of managing the performance of workloads and the absence of computer assistance at

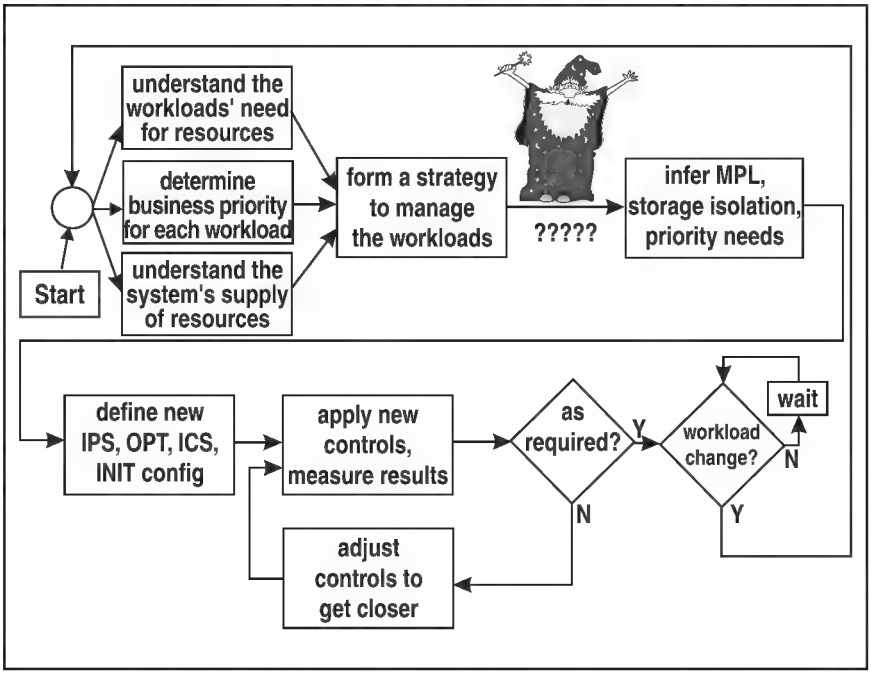


Figure 6-2. Workload Management in Compatibility Mode

critical steps in the process make it clear that the process and its underlying model is itself flawed. Consider the flowchart in Figure 6.2.

The crucial step, getting from the workload management strategy to a set of simple statements that can be rendered into the driving parameters of SRM, is mysterious and uncertain. The wizard (or some other form of supernatural intervention) might very well help. Also, the measurements are not part of MVS and the feedback steps are manual and outside the scope of SRM—at the very point at which computer measurement and analysis would help, they are external to the system.

As far as SRM without the Workload Manager is concerned, it meets the objectives of its name: it manages system resources. It knows *nothing* about workloads. That's not good enough, and that's why something better was needed. SRM with Workload Manager in goal mode covers more of the flow chart with the full power of the System/390.

6.7. Summary

In this chapter, we've examined some basic SRM concepts, including how common notions of time are given special meaning in an SRM context. We've also examined service units, service rates, and the way that service rates affect swapping decisions in MVS/ESA SP 4.2 and later releases.

6.8. Chapter Questions

1. Look at RMF or CMF Workload Activity Reports for your system at a peak time of day when all workloads are running well. How much of the service to your key workloads is due to MSO service units?
2. Repeat the procedure of question 1 for an off-peak time of day and compare the relative contribution of MSO for the same workloads at the two times. How much use is made of the service unit totals in your installation? Does the variability of MSO contribute to inappropriate decisions?
3. Calculate the peak-hour service rates for your most important workloads with MSO included and recalculate them without MSO. Save these numbers; they will be needed after you finish reading Chapter 7.

4. If you are using a pre-SP 4.2 system, examine the **OBJ**s in your IPS to determine the range of service rates in them. Then compare those service rates with the service rates you found by examining the Workload Activity Reports. Are they consistent? How should the rates in the **OBJ**s be adjusted? (The same consistency test may be applied to service rates in ASRVs or DSRVs.) In a 4.2 system, look at the service rates received by the domains and compare them with the rates reported in the Workload Activity Report. Are the domains operating within their specified service rate limits at peak hours? How should the IPS be changed to make this so?
5. Have you taken any actions to defend your system against central storage constraint? How should they be changed given the presence in SP 4.2 of the Working Set Manager?

Chapter

7

The Workload Manager in Goal Mode

Even though the System Resources Manager was revolutionary in its time, SRM did little to help performance analysts and system programmers manage the performance of real workloads. It wasn't SRM's fault. SRM did a very good job of doing just what its name connotes—managing the distribution of system resources to workloads according to the cryptic directives in its controlling parameter sets. The original SRM designers did not take on the challenge of managing workload performance. Eventually, however, they did recognize the workload management problem and introduced a new mode of operation for the SRM. MVS Version 5 introduced the Workload Manager and a new approach in workload performance management.

7.1. SRM and the Workload Management Problem

We begin looking at Workload Manager through the familiar technique of reviewing what came before it. The fundamental problem of attempting to use SRM to manage the performance of workloads is that SRM didn't do anything to achieve that end. Consider these observations regarding workload performance management in MVS:

- **MVS is a system that runs well when fully loaded**

  At least it *can* run fully loaded. However, performance may suffer if the work is homogeneous. For instance, if the en-

tire aggregate workload on a uniprocessor is short-running CICS transactions of equal business importance, significant internal CPU queuing appears in CICS at about 70 to 75 percent CPU utilization.

When there is other work that can meet its business objectives with a lower dispatching priority than that of CICS, and that work does not cause contention for resources used by CICS, it is possible to fill the system with such low-priority work without significant impact on the more important workload.

- **It's thus best to load the system with different kinds of work**

  An assortment of work with complementary resource usage patterns and little intrinsic contention allows full use of all resources. This idea was perceived by IBM's engineers back in the '50s when they invented the I/O channel and made multiprogramming possible.

- **Tuning trade-offs are possible with a mix of work**

  "Know who your *Loved Ones* are... and always have someone else to kick around!"[1] The opposite case, that of an aggregate workload that is homogeneous in type and execution characteristics and of uniform importance to the business, *cannot be tuned*! Of course, such a workload can be optimized within its environment but tuning—solving short-term performance problems by redistributing resources—requires winners and losers.

- **Each kind of work has a service target**

  Whether you have formal service level targets or agreements or none at all, those who make up a user community have expectations for service: response times for interactive environments and turnaround time for batch (including printed output to its delivery point). Committed targets provide an objective basis for evaluation, but the unstated targets based on user expectation alone are as real as formally committed ones. A further benefit of having committed targets is to have an answer to the question, "How do you know when you're doing well enough?" There is value in knowing when all targets are being met, so that tuning

1 I remember first hearing this from Siebo Friesenborg of the IBM Washington Systems Center, sometime around 1980.

effort may be turned to the potentially more rewarding task of identifying applications that are performing poorly for intrinsic reasons, and tuning them.

- **Most work runs well in MVS with little tuning**

  The default MVS system will run a mixture of work such as TSO, CICS, and batch with respectable performance—until a resource shortage, particularly a central storage shortage, arises. At that point the default system behavior can produce unanticipated and inconsistent results. Basic system tuning can soften the default behavior and make it more predictable.

- **There are losers—some work needs special handling**

  There is a class of work known as *numerically intensive computing* or *NIC*. This work, usually found in scientific or engineering disciplines, typically has both high CPU demand and a widespread and variable "footprint" in central storage. The typical form of special handling is to isolate the NIC work in special job classes that are allowed to run at hours when the ill-behaved programs cannot cause significant damage to more response-time-oriented work.

  When NIC is an important workload, this shunting aside of problem jobs may no longer be acceptable. Instead, there is a severe challenge to the support staff to manage the system so that the NIC work runs reasonably well without damaging online service. Unfortunately, MVS in the past provided no tools to deal with such a balancing act. This problem went unsolved until the introduction of Working Set Management in MVS/ESA SP 4.2.

  However, Working Set Management as implemented in SP 4.2 solves only part of the problem. It does well at optimizing the use of central storage, managing unproductive use of the CPU to acceptable levels, and ensuring that ill-behaved address spaces will receive some level of service. Unfortunately, the Working Set Management approach, when guided by these objectives, does not help the ill-behaved work unit that is also important. It would take the changed point of view that came with goal mode to allow Working Set Management to reach its full potential.

- **The system must be managed to meet the targets**

  The task is simply to allocate resources to the various kinds of work units in the system in such a way that each workload's performance targets will be met. The tools at hand include the hardware configuration, SRM's controlling parameters, initiator settings, and the tuning parameters of the subsystems such as IMS and CICS.

  If SRM did perfectly all that it can do, there is no guarantee that workloads would meet their service targets, unless the driving parameters of SRM, by some lucky accident, were so well matched to the workload characteristics, the system configuration, and the required service levels and business priorities that the service levels were met consistently over time.

- **Too many good things are optional**

  With its large number of external parameters, SRM is overly dependent on the experience level of those in charge. Almost everything added to SRM over the years, with the notable exception of Working Set Management, required research, analysis, and the will to take a risk and try something new in order for its benefits to be realized.

- **The system knows nothing about workloads**

  Workload management has been indirect at best. It is unreasonable to expect SRM to do more than its name indicates. SRM does not know workloads and their needs, and no small change could have given it that knowledge.

7.2. Introducing the Workload Manager

The IBM response to the complexity of workload service management was to go well beyond the resource-oriented management done by SRM and change the orientation to that of defining service goals and managing the system so that the goals were met if they could be.

7.2.1. Why was it needed?

Two powerful needs are met by Workload Manager.[2] The first is the need for a response to the limitations of prior SRM imple-

2 From this point on, mention of Workload Manager or WLM means Workload Manager goal mode unless specific mention is made of compatibility mode.

mentations, summarized above. The second and perhaps the driving need from an IBM justification point of view, was the requirement to simplify the management of the parallel sysplex. Without a global workload manager across the sysplex, the complexity of service delivery management across many independent MVS images could have rendered parallel sysplex unacceptable.

7.2.2. Prototypes for Success

The best inventions make use of, and often redirect, the successes of the past. Some ideas that carried forward into WLM include:

- **algorithmic adjustment of system resource controls based on measurement**: As reflected in Working Set Management, this approach proved very successful. It was redirected in two ways: by adding management to service goals to the original resource-oriented goals of the storage management algorithms, and by broadening the approach to extend to the management of other resources.
- **controls that work automatically without external parameters**: This too was a lesson of Working Set Management. It was extended to cover all the resources that SRM deals with. In goal mode, there are no parameters, just policies.
- **tuning based on execution-state analysis**: The data collected by performance monitors like RMF Monitor III, OMEGAMON II for MVS, and similar products provides a basis for what Candle Corporation has long referred to as "The Logical Tuning Approach"—a simple set of steps based on service targets and execution-state data:
  - determine if there's a problem (if not, come back later)
  - identify the source of delay[3]
  - if possible, take the delaying resource away from the undelayed unworthy address space(s) and make it available to the Loved One in pain

  The transmutation of this process was to make the data collection standard, continuous, and efficient, by integrating the measurements into the lowest-level parts of MVS.

3 As Ray Wicks of the IBM Washington Systems Center often says, "Where's the pain?"

- **service level agreements**: IBM transformed this external service management tool into the prime control over the management of system resources.

7.2.3. How it Works

Workload Manager needs the following elements in order to succeed:

- a *service policy* that classifies work according to installation-defined criteria into *service classes* that reflect the service requirement (*goal*) for each subdivision of the aggregate workload. The service policy also includes, for each service class (or *performance period* of a multiperiod service class) an *importance* level reflecting the installation's priority for that subset of the work. The service classes are aggregated into *workloads* for reporting and resource accounting. *Report* classes provide additional granularity or alternative structures in reporting and resource tracking.
- MVS services to collect data that allow WLM to assess whether each service class period's goal is being met. Periodically, WLM compares the actual performance against goal and calculates a *performance index* or PI. The PI is a normalized measure of the extent to which the goal is being met. Values greater than 1 represent goals not met; a value less than 1 represents a performance better than the goal, and a value of exactly 1 means that the goal is exactly met. This information allows the analysis routines to determine which address spaces need help, and which are doing well enough so that they might become *donor*s of resources.
- MVS services to collect data that characterizes the execution and delay states of each executing unit of work. Later decisions depend on this information, to identify the delays that might be reducible if additional resources were to be made available to work units through the corresponding address spaces—and to avoid trying to help where no help is needed.
- a set of MVS services to change the operating conditions of work units by adjusting the availability of resources to address spaces such that as many goals as possible are met, starting with the most important work and moving down to the least important. The crucial, sophisticated algo-

rithms are found here. Advanced techniques such as historical and projective modeling are used, extending the algorithmic analysis and cost-benefit tradeoff analysis done in Working Set Management. Actions and results are tracked, and unproductive choices may be reversed and temporarily removed from consideration on behalf of a particular work unit.

Workload Manager manages all the resources that were previously manipulated through SRM parameters, including dispatching priority, storage isolation or protection, swapping, and page stealing. Beginning with OS/390 Release 3, WLM also reorders the IOS queues—the queues of I/O requests waiting for busy devices—as another means of attempting to meet goals. Because these system "knobs" are under WLM control, they are not available to be set by programmers. For descriptions of page stealing and swapping that remain reasonably valid in goal mode, see Appendixes A and B. References to SRM-initiated actions should be disregarded in favor of the goal-oriented actions described in this chapter.

7.3. Creating the Service Definition

In contrast with compatibility mode's three sets of SRM parameters in SYS1.PARMLIB, all of the control information for goal mode is in one place.[4] The name of the data set isn't important; it is an XCF Couple Data Set and is not an editable plain text file. There is a specific interface to it through the ISPF dialog invoked through a REXX EXEC called `IWMARINO`, available to anyone for browsing the service definition.[5] Loading the service definition to the couple data set or activating a policy should require authorization, of course. The operational details are left to the system programmer.[6]

4 The `IEAICSxx` and `IEAIPSxx` members are ignored. `IEAOPTxx` remains with those parameters that do not relate to MPL adjustment, swap controls, or expanded storage criteria. Those remaining are `CNTCLIST`, `CPENABLE`, `DVIO`, `ERV`, `MCCAxCTH`, `MCCFXxPR`, `RCCFXxT`, and `RMPTTOM`.These are described in Appendix B.

5 The EXEC is distributed as a member of a data set called `SYS1.SBLSCLIO`.

6 For authoritative information regarding the details of Workload Manager, including setting up the service definition, see the IBM publication *MVS Planning: Workload Management*. The publication number for the OS/390 version is GC28-1761; for MVS/ESA SP Version 5 it is GC28-1493. Each of these publications also has cross-references to other books needed in system setup.

7.3.1. Service definitions and service policies

There is exactly one service definition in a sysplex's Workload Manager Couple Data Set, made up of at least one service policy. Only one service policy may be active at a time. The definition and its policies may be exported and imported, and since MVS/ESA SP 5.2.2 there has been an IBM-supported Application Programming Interface (API) to the service definition. Although no one has announced exploitation of the API to date, its existence facilitates the development of tools that can query and manipulate the policy, or perhaps provide alternative user interfaces for definition.

Creating a new definition is not a straightforward process at first glance. The ISPF application will not permit anything to be defined if the objects to which it makes reference are not already defined. It turns out that one can simply work in the order of the primary menu that appears after the "create new definition" selection is made[7]. A possible order of tasks would then be:

1. Set the service definition coefficients.

Although the defaults of `CPU=10.0`, `SRB=10.0`, `IOC=5.0`, and `MSO=3.0` are carried forward from compatibility mode, there is good reason to change the values. Chapter 6 detailed why MSO should be set to 0. If there is no reason to preserve continuity of coefficients, the simplest setting for the other coefficients is to make them all equal to 1.0. By doing so you can avoid encountering excessively large service accumulations and service rates on larger systems. There is also a specific goal-mode-related reason for doing so: the service rates used in defining resource groups are unweighted CPU (TCB) and SRB service rates. The unity coefficients put ordinary service rates on the same footing as resource groups' service rates.

2. Name the service policy and provide a text description.

That's all there is to this step. Content will be added by the subsequent steps.

3. Name the workloads and describe them.

Every work unit that enters the system must go into exactly one service class, which is in turn associated with a single workload.

7 One change to the order reinforces the top-down approach: we place the specification of Service Definition Coefficients first.

Within that constraint, workloads may be structured in many ways. The two usual possibilities are to group by subsystem type (e.g., all CICS transactions in one workload and all batch jobs in another) or by business organization, with, for instance, all CICS Accounts Payable transactions as well as the batch check-writing jobs in the same workload. Bear in mind that each work unit may also be classified to a single optional report class, so a second organizing scheme may be created for reporting purposes.

4. Name, describe, and define any needed resource groups.

The idea of resource groups is somewhat contrary to the idea of goal-oriented workload performance management. However, it's important in some installations to limit the amount of service received by a workload. In some installations as well it may be part of a service agreement or a contractual obligation to guarantee the minimum amount of service to be received by a workload.

A resource group is used to manage (limit or guarantee) the service received by one or more service classes. Each service class may be associated with at most one resource group. The unit for the maximum and minimum specifications is unweighted CPU service units, consumed in both TCB and SRB modes, for all systems in the sysplex that are running in goal mode. Because transactions in subsystems such as IMS and CICS are not directly associated with CPU service, resource groups may not be specified for service classes to which such work units have been classified. However, resource groups may be applied to the service classes representing the address spaces which serve the transaction workload. In other words, resource groups may be associated only with service classes populated by address spaces, not with abstract transaction workloads. There may be a maximum of 32 resource groups in a service definition.

The maximum service rate specification of a resource group has a simple effect, and only on the service class(es) associated with the resource group. If the aggregate service rate received by the address spaces in the service classes tied to the resource group is exceeded, SRM takes action to "throttle" the service received by those address spaces, across the entire sysplex. The mechanisms employed include swapping (if possible), dispatching priority adjustment, and "capping," a form of time slicing that intersperses slices of nondispatchable time with those of dispatchable time. Exceeding the maximum service rate may

cause the service class's goal to be missed. This condition is noted in the reporting information for the service class. There is no default value for the maximum service rate.

When a minimum service rate exceeding the default value of zero is specified for a resource group, it may or may not be reached. As IBM says in *MVS Planning: Workload Management*,

> If a service class with a goal is assigned to a resource group with a minimum specified, and there is insufficient work, the goal may be overachieved....
>
> The minimum is guaranteed if sufficient work exists, and other work's goals are not impacted. WLM gives more resource to meet the minimum capacity, even if all work in the resource group is achieving its goals. If there is a resource group defined for a service class with a discretionary goal, workload management achieves the minimum as long as the goals of work running in any other service class are not impacted. If other performance goals are impacted, then workload management does not maintain the minimum.

It is not completely clear from the foregoing that work with a nondiscretionary goal and an excessive minimum resource group will *not* cause other goals to be missed. If you choose to establish a minimum service rate in a resource group, understand the possible consequences and make the minimum rate as low as possible. Another possible unintended consequence is that "promotion" of discretionary work and overachievement of goals may create inappropriate expectations for the performance of affected workloads.

5. Name, describe, and define the service classes.

Service classes contain goals and name-references to workloads and (optional) resource groups. Multiple periods may be defined for a service class. In this case, the duration of each period other than the last must be specified; the unit is total (weighted) service units.

Considerations for setting goals appear below. Work is placed in a service class as work units arrive, through the operation of classification rules.

Although up to 255 service classes[8] may be defined, the number active at any time should be limited. IBM's usual recommendation is to keep the active total under 25.

6. Name, describe, and define classification groups.

Many classification rules depend on sets of generic or enumerated names, to allow precise selection of which particular work units go to which service classes. Instead of defining a rule for each name of such a set, a name group of the appropriate type allows the rule to be defined once, referencing the name group instead of a name. Depending on the subsystem type, a group can contain names based on connection type, logical unit (LU), net ID, package name, plan name, subsystem instance, transaction class, transaction name, or user ID. Although name groups are not needed, defining a set of names as an object and referring to it by its name can reduce clutter in the classification rules and is in some cases more efficient than requiring numerous single-name match tests.

7. Name, describe, and define classification rules.

This must be done for each active subsystem and optionally for subsets of work units within each subsystem. Since OS/390 Release 1, the supported subsystems are: ASCH, CICS, DB2, DDF, IMS, JES, OMVS, SOM, STC, and TSO. The variables or name types on which rules can be defined vary from subsystem to subsystem. Valid rule types for each subsystem are shown in the following table.

		Subsystem									
Code	Rule Type	ASCH	CICS	DB2	DDF	IMS	JES	OMV	SOM	STC	TSO
AI	Accounting Information	X		X	X		X	X		X	X
CI	Correlation Information			X	X						
CN	Collection Name			X	X						
CT	Connection Type			X	X						
CTG	Connection Type Group			X	X						
LU	LU Name		X	X	X	X					
LUG	LU Name Group		X	X	X	X					
NET	Net ID			X	X	X					

8 This number drops to 100 in OS/390 Release 3. At the same time, the maximum number of report classes rises to 999.

		Subsystem									
Code	Rule Type	ASCH	CICS	DB2	DDF	IMS	JES	OMVS	SOM	STC	TSO
NETG	Net ID Group			X	X	X					
PK	Package Name			X	X						
PKG	Package Name Group			X	X						
PN	Plan Name			X	X						
SI	Subsystem Instance	X	X			X	X				
SIG	Subsystem Instance Group	X	X			X	X				
SPM	Subsystem Parameter								X		
TC	Transaction Class	X				X	X				
TCG	Transaction Class Group	X				X	X				
TN	Transaction Name	X	X			X	X			X	
TNG	Transaction Name Group	X	X			X	X			X	
UI	UserID	X	X			X	X	X		X	X
UIG	UserID Group	X	X			X	X	X		X	X

Note that the "SPM" rule type and only that type is used exclusively for the SOM subsystem type. The design permits any subsystem to use that type but no other has done so to date.

In contrast with the fixed order of rule evaluation performed in the ICS of compatibility mode, the order of rule evaluation in goal mode classification can be specified explicitly. The only exception is subsystem type; it always comes first.

Each of the rule types provides the test value or values (in the case of a group rule type) for a series of comparisons in the order specified in the classification rules for the subsystem type. Rules form a tree structure with the root at the subsystem default and are defined with numerical levels. At a given level there can be different rule types; however, a given rule type may exist only once in each branch. Once a rule produces a match, succeeding rules at the same level are ignored.

The last successful match in a hierarchy sequence concludes the process and the corresponding class is assigned. A report class may be assigned at a different level, either higher or lower, in the hierarchy than the service class.

Each subsystem type except for STC should be defined with at least a subsystem default service class. The STC subsystem requires a different approach to definition so that selected STCs

may be directed to special system-defined service classes. The first example below shows the special treatment for setting up the STC subsystem.

Special considerations for the STC subsystem. This example shows how the STC subsystem could be classified. Details may differ but the essential elements are the omission of a default service class and the level-1 catchall at the end of the rule set. This treatment is necessary to accomplish the fol-

```
Subsystem Type . . . . . : STC
Description  . . . . . . . . . . All started tasks

  -------Qualifier-------------                -------Class--------
   Type       Name      Start                   Service      Report
                                    DEFAULTS:  ________     ________
1  TNG        STC_HI    ___                     ________     ________
1  TNG        STC_MED   ___                     SCVMED__     ________
1  TN         *         ___                     SCDISCR_     ________
```

lowing specific goals:

- The deliberate omission of a default service class ensures that system address spaces such as GRS will not be classified. Doing so avoids having these address spaces taken out of the special SYSTEM service class. (See section 7.3.2 for a discussion of the system-specified service classes.)
- The first rule allows specific started tasks to be assigned to the second system-defined name, SYSSTC. This is done by having a blank specification for both service class and report class in the first transaction-name-group rule.
- Provide for a service class with an intermediate goal for important but not highest-priority STCs. Explicitly named address spaces in the second name group are assigned to SCVMED.
- Make sure that *unspecified* STCs do not receive an undeservedly favorable goal by allowing them to fall through to SCDISCR, presumably a service class with a discretionary goal.

The only drawback of this kind of classification is the need to review the name lists periodically to make sure that newly added

STCs or those whose importance or use has changed are in the appropriate lists.

Classification of non-STC subsystems. The example below illustrates the use of classification rules for a transaction-based subsystem type, in this case CICS at Version 4 or later. (The subsystem is a set of CICS transactions, not the set of address spaces that provides service to the workload.)

In this oversimplified example, CICS transactions that do not match any of the specific rules are assigned to service class

```
Subsystem Type . . . . . : CICS
Description  . . . . . . . . . . CICS subsystem
                                                  -------Class--------
                                                  Service      Report
                                        DEFAULTS: CICSSC3      CICSOTHR
1  UIG          APC                               CICSSC1      ACCTPAYC
2    TNG          VENDOR                          ________     VENDOR
2    TNG          GOVT                            ________     TAXES
3      SI           CICSEXP                       ________     EXPRESS
1  UIG          PURCH                             CICSSC2      PURCHASE
2    TNG          VENDORA                         ________     A_LIST
2    TNG          VENDORB                         CICSSC3_     B_LIST
```

CICSSC3 and report class CICSOTHR. These are any transactions not associated with a user-ID in the name lists APC or PURCH, since those are the only first-level rules. All transactions from the APC users go to service class CICSSC1. Those in the transaction-name-group VENDOR go to report class VENDOR. In the transaction-name-group GOVT, all transactions go to the report class TAXES, except those entering through CICS subsystem name CICSEXP; those are assigned to report class EXPRESS.

In the second user-ID group, note that the subsystem's default service class is explicitly assigned to the second transaction-name-group.

In a real CICS or IMS transaction processing workload,[9] it is very desirable to segregate transactions of similar importance and performance profile into separate service classes. To make sure that WLM's adjustment of access to resources affects only the "deserving" transactions, those transactions of each service class should also be segregated in separate server address

9 Transactions are classified only if the subsystems are at the appropriate Workload Manager-enabled release levels; CICS Version 4 or later, IMS Version 5 or later.

spaces (AORs or MPRs) that are consistent with the service classes.

Such a breakdown is needed to avoid the "free ride" phenomenon. Suppose (in the example above) that the high-priority transactions in CICSSC1 are in the same AOR as the low-priority transactions in CICSSC3. If the APC-submitted transactions are performing slower than goal, WLM might increase the dispatching priority or storage protection of the serving AOR address space. As a result, the VENDORB transactions would benefit from the improved resource availability. WLM would be powerless to treat that workload as a resource donor because of the high-performance transactions being served by the same address space. If the transactions classified into CICSSC1, CICSSC2, and CICSSC3 were respectively in different AORs, the resources could be directed only to the service class that needed help, and the less important workload's server TOR could then become a resource donor.

7.3.2. The default service policy

SYSTEM and SYSSTC are special service classes with "system goals" having high fixed dispatching priorities. Together with SYSOTHER, a service class with a discretionary goal, they comprise the underlying wired-in, default service policy. The favorable service classes are always active and should always be in use for the key address spaces that constitute the MVS infrastructure. SYSOTHER should normally never be seen, but it may appear if a previously unused subsystem is activated. For instance the first time that OpenEdition is fired up, OMVS address spaces may show up as SYSOTHER to the dismay of the users.

If work units show up in SYSOTHER, that should be a signal to update the active service policy and add appropriate classification rules and service classes for the newly active workload.

There have been reports that some sites have operated "successfully" with no defined service policy.[10] Doing so is somewhat like running MVS in compatibility mode with no IPS or OPT. In fact, one way of ending up in goal mode with no service definition is to bring up a system in Version 5 or OS/390 without an `IPS=xx` parameter in the IEASYS00 member of SYS1.PARMLIB.[11] The

10 The "success" was based on having a generous excess of resources.

11 The meaning of omitting this parameter changed in Version 5. Through Version 4, it meant "use IEAIPS00, the default." Starting with Version 5, it means

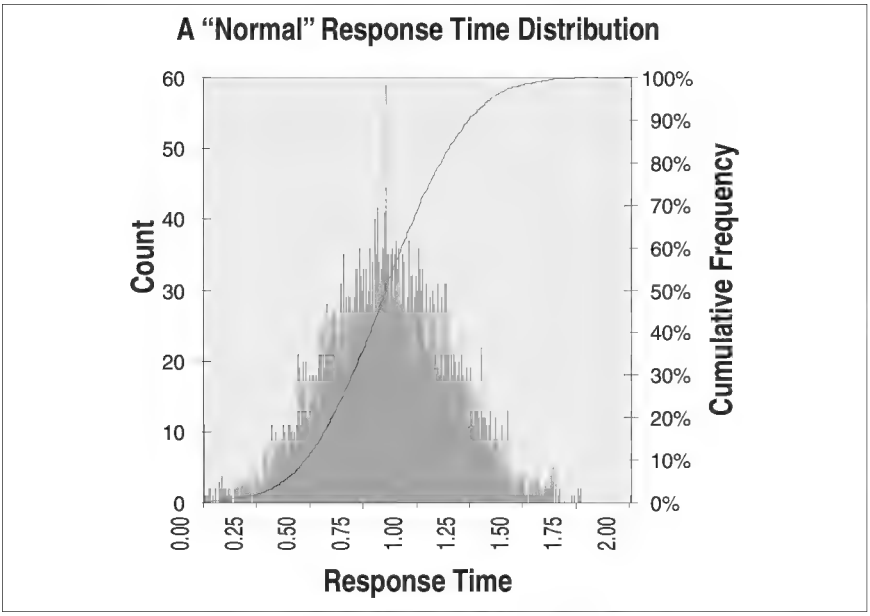


Figure 7-1. A Normal Distribution of Response Times.

system will run reasonably well until there is a resource constraint. At that point there will be unpredictable degradation of seemingly random work units as the discretionary address spaces are swapped out, dropped to bottom dispatching priorities, or capped.

It's much better to create even the most rudimentary service definition. A good head start, well beyond rudimentary, may be found in recent CMG Proceedings (UKCMG 1996 and CMG 1996) in the form of Cheryl Watson's *"Quickstart" Service Policy.*[12] Cheryl has kindly provided permission to present the summary charts of her paper at the end of this chapter. The rationale for her recommendations and further details may be found in her paper or in her newsletter.

"initialize the system in goal mode."

12 The quickstart policy was originally published in *Cheryl Watson's Tuning Letter.*

7.3.3. Setting goals

Response Time Goals

The response time goal, whether it is an average or a value at a percentile, is measured against internal response time only. If at some future time WLM could become capable of including consideration of network response time, it is unlikely that the same goal definition would apply in that case.

When choosing which kind of response time goal to use, some advice-givers seem to ignore an obvious consideration—what is the nature of the workload? Instead, they tend to rank the types of goals, regarding the goal of response time at a percentile as just a more stringent variation of the average response time goal. In fact it usually is not.

For example, consider a response time distribution such as that shown in Figure 7-1. It is a good approximation to a normal distribution with a mean of 0.85 second and a standard deviation of 0.3 second. This is the kind of distribution for which "average response time" has a valid meaning. A normal distribution is described completely by two parameters, mean and standard deviation, usually denoted as μ (mu) and σ (sigma). (The vertical scale is irrelevant.)

In a normal distribution, percentile values may be calculated from the parameters. The 90th percentile is approximated by $\mu+\sigma$, the 95th by $\mu+2\sigma$, and the 98th by $\mu+3\sigma$. This sounds easy, but unfortunately it's too good to be true. Consider a couple of characteristics of the normal distribution:

- both upward and downward deviations from the mean are equally likely
- small deviations are much more likely than larger ones

A moment's thought makes it clear that the normal distribution may be a good model for the distribution of 11-year-old boys' shoe sizes, or the heights of adult females, but is a thoroughly inappropriate model for transaction response times. Typical TSO, APPC, CICS, IMS, or OpenEdition transactions may be characterized by the following observations:

- Transactions are not uniform in their resource demand. The widest variability shows up in unstructured environments such as TSO, the least for specific groups of transac-

tions in a carefully managed transaction manager environment such as IMS or CICS.

- For each type of transaction there is a lower limit to response time, consisting of the CPU execution time needed when there is no delay of any kind. The CPU cannot go any faster.
- Each class of delay persists for at least a nonzero minimum time. For instance the minimum resolution time for a page fault is determined by the kind of storage on which the page resides and the path length through the various MVS resource managers needed for the particular resolution actions.
- Some delays can occur more than once in the lifetime of a transaction, and others can persist for multiples of the minimum time.
- The probability of each delay is variable, subject to a large number of factors random with respect to the transaction.

Populations of transactions with these characteristics do not fall into any of the common statistical distribution models. Figure 7-2 represents such a distribution for composite transaction profiles. The combination of intrinsic variability, one-sided varia-

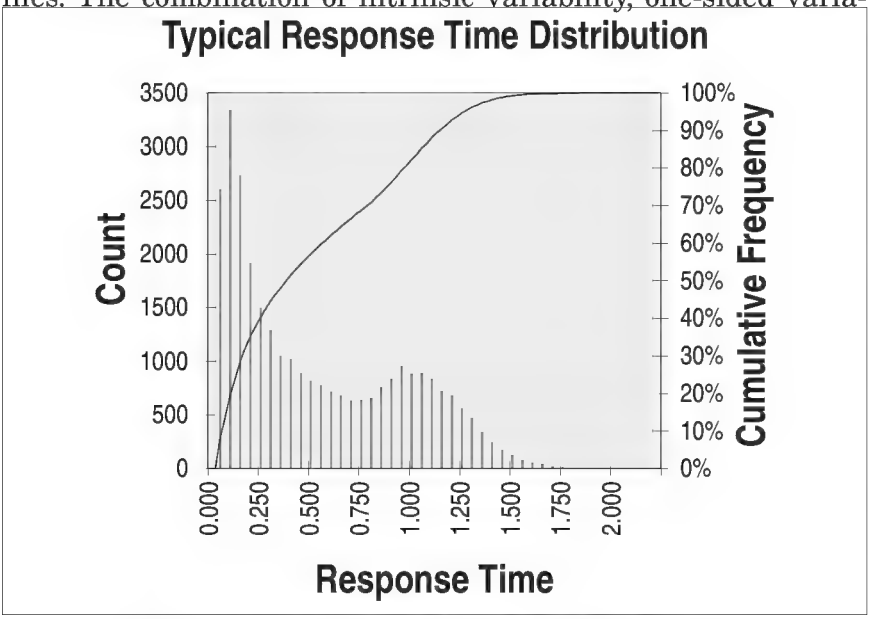


Figure 7-2. A Typical Distribution of Response Times.

tion, and the quasi-discrete nature of many delay factors makes it necessary to fall back on nonparametric statistics. In effect we are reduced to sorting completion times into bins, each representing a small range of time, and counting the contents of the bins at some summarization interval. If we then plot the running sum of the bins starting with the first interval, normalizing the count axis to 100 percent[13], we can read off response time at a percentile, without calculating misleading statistics like the mean and meaningless numbers like the "sample standard deviation."[14] This is the basis of the process used for percentile calculations within Workload Manager.

Choosing which kind of response time goal to use should take into account the characteristics of the workload.

The nonparametric model is general, safe, and appropriate even in the rare case of a workload that fits the normal distribution. In the case of such a workload, it's important to recognize the relationships between the mean and percentiles that may be of interest. Since the normal distribution is symmetric, the mean is also the median—the 50th percentile. An average response time goal is thus roughly equivalent to the same response time at the 50th percentile. To get to the 90th percentile, the mean must be augmented by the standard deviation. In the case of Figure 7-1, an average response time goal of .85 second is equivalent to a goal of about 1.15 seconds at the 90th percentile.

For non-normal distributions, no such relationship exists. One should create a cumulative frequency distribution plot to establish the 85th or 90th percentile response time—or just ask the users what response time they perceive as "typical." It will usually be somewhere around the 85th percentile of their experience.

The response time definition is a little different for batch jobs. It includes time spent on reader queues. Although that variant seems intuitively sound, the queue time for deliberately held jobs can corrupt the response time numbers. On the other hand, excessive time on nonheld queues can provide a direct indication of a need to revise initiator settings. Batch jobs that are likely to be held should not have response time goals—at least not the same goals as jobs that are not likely to be held. If the

13 We have thus created a Cumulative Frequency Distribution (CFD) plot.

14 Means and standard deviations are valid and useful in the analysis of quantitative data that is stable and unimodal. Utilization data for homogeneous time intervals is such an example.

classification for batch is done on the basis of job classes, held and nonheld classes should go into different service classes. It's also not effective to have a response time goal in a service class [period] with a low rate of transaction completions—fewer than about one per minute. The better alternative for such work units is to use a velocity goal, as described in the next section.

If the nonheld batch jobs follow a common pattern, with far more short-running jobs than long-running ones, a single service class with multiple periods might work. The first period might have a duration of 10,000 service units with a response time goal of 1 minute for 80 percent of transactions. Second period could be 50,000 service units at a response time of 10 minutes for 80 percent, and the final period could have a velocity goal of 10 percent. Importance levels would be dependent on business priorities. If the work likely to fall into last period is of very low business importance, a discretionary goal might be appropriate.

7.3.4. Velocity goals

Velocity goals should be used for unending started tasks with a lower performance requirement than those in the SYSTEM or SYSSTC service classes (see §7.3.2), for long-running batch, and for other address spaces not easily tied to response time measurements. Velocity seems simple: express as a percentage just the amount of time spent using the CPU, divided by the sum of that time and all the delays for CPU and storage. "Storage" delays include MPL, paging, and swapping delays.

In OS/390 Release 3, I/O queue delay (IOSQ plus PEND) is added to the denominator, and I/O active time (connect plus disconnect) is added to both numerator and denominator. Since I/O time can be a considerable part of total time, the Release 3 changes may have a significant effect on observed velocities. When going to OS/390 Release 3 or later, it may be necessary to adjust existing velocity settings depending on new observations.[15] It is also important to recognize that both connect and disconnect time are tunable components of I/O activity, but

15 To ease the transition to goal-oriented management of I/O priority, the enhanced definition of velocity and the reordering of IOS queues is an option in OS/390 Release 3. The option is selected on the ISPF panel on which the service definition coefficients are specified. The default value of the option is **OFF**.

Workload Manager does not address these components. I/O analysis and tuning is covered in Chapter 9.

It is easy to set velocity goals that are unattainable, so it might be prudent to get an idea of what velocities are likely to be attained. The RMF Workload Activity Report in RMF Version 5 shows velocities by performance group and domain for systems running in compatibility mode. Other performance measurement products can provide similar information. If a particular workload running satisfactorily at a peak time shows an attained velocity of 40 percent, that value (or five to ten percent less) might be a good choice for the corresponding goal in goal mode. Setting a more demanding goal might be unwise. A demanding velocity goal might be 70 percent; ordinary batch might do acceptably well at 10 or 15 percent. Velocity steps of less than 10 percent between different goals might turn out to be meaningless.

Note that the idea of velocity and how it is defined may change yet again in subsequent releases of OS/390. As always, make sure to review IBM announcements and the latest updates to the IBM documentation covering WLM before establishing a new service definition.

7.3.5. Discretionary work

The discretionary "goal" is not really a goal; it's the absence of a goal. The plain meaning of "discretionary" is "subject to discretion or choice." In other words, WLM is free to choose whether to run discretionary work units or not, and if it runs such work, which of those work units to choose. If we recall the underlying imperatives of SRM, the practical effect is that discretionary work will be run to the limit of available resources. In goal mode, that imperative is tempered by the additional concern that all goals that can be met will be met.

When there are sufficient available resources (beyond those necessary to meet all goals) over a reasonably long interval, all discretionary work will run in no particular order and without reproducible performance. If sufficient resources are not available, the completion times for some discretionary work may be severely elongated in no particular pattern.

The discretionary classification has one unique benefit: multiple discretionary work units run in a single mean-time-to-wait group. Within that group, dispatching priority order is determined by the

average interval between voluntary waits (e.g., for I/O completions). Those address spaces with the shortest wait intervals have the highest priority within the group. This arrangement yields maximum throughput with a mixture of discretionary work of different balance between CPU and I/O content.

Deciding which work goes into service classes with discretionary treatment depends on these considerations:

- What work units fall into the genuinely discretionary category given the current system constraints? If there is a category of batch work, test subsystems, and last-period TSO/APPC/OMVS work for which there is no business value associated with undelayed service, these are the candidates. Even so, it might be more orderly to run some of the work at a low velocity (five percent or less) and lowest importance.

- What portion of the system or sysplex resources are consumed by the entire aggregate workload? If central storage is unconstrained and less than 80 percent of the CPU is consumed, then placing 25 percent of the workload in the discretionary category might be perfectly all right. All the work would be processed without undue delay.

  As the committed resource level or degree of resource constraint increases, the amount of work in discretionary should be reduced. More critical judgment of business importance might be necessary—or there might be a sufficient level of service complaints to guide the decisions.

  WLM is unduly constrained if there is no discretionary work. A reasonable minimum level might be 5 to 10 percent of total delivered service units at peak time.

After a period of experience, the division between truly discretionary work and work with very humble goals might become more apparent. It might be amazing to see how previously unknown business goals can suddenly surface.

7.3.6. Choosing importance levels

There are five importance levels, designated from 1 (highest) to 5 (lowest). The importance specification provides a previously-unavailable second dimension in performance management. It's no longer necessary to specify an unrealistically high dispatching priority to make sure that an important workload is

not affected by other work. Importance comes into play only when a goal is not being met.

It's entirely reasonable to have a service class period with a goal of one second response time at the 85th percentile and importance 4, while another has a goal of 5 second response time at the 90th percentile with importance 2. As long as all the goals at higher importance levels are met, WLM will strive to meet the one-second goal. However, if the importance 2 goal is not being met, and the work at importance 4 has a resource that can help, meeting its goal may be sacrificed so that the higher-importance goal can be met.

In short, pick importance levels to create an order in which goals will be sacrificed as resources become constrained. This order should closely approximate actual importance to the business or institution. A high importance level will not improve performance beyond what the goal requires, but will ensure that the goal is met more frequently than it would be with a lower importance.

7.4. Measurement and Monitoring

One hidden advantage of goal mode is that most of the data that would need to be collected for performance monitoring is already collected by the system. Because WLM at this time does not explore the details of ENQueue delays, or of I/O delays at the device level, such information would need to be collected by a comprehensive performance monitor. Chapter 9 covers measurement and prediction in any MVS environment, with emphasis on goal mode. However it is much more nearly true in goal mode than in compatibility mode that performance monitoring of workloads can revert to a standby capability rather than being the center of systems management. Instead, the focus of performance monitoring and systems management can shift back to a concern for availability—of the hardware, the network, the connections, and the applications. Effort formerly spent trying to solve the puzzle of workload performance management with the compatibility mode SRM can be applied, with greater return on investment, to the tuning of key applications.

7.5. Cautions, Limitations, and Extrapolations

Managing the performance of MVS workloads in goal mode is so easy that one may be tempted to ask "Is that all there is?" For

those who feel that there is some loss of control in going to goal mode, we might point out that the resources are all still there, but managing them has become less of a front-line system programmer activity and more indirect. For those who insist on understanding what SRM does at a low level, WLM can upon request write a very detailed set of SMF records, grouped as Type 99. The documentation of these records provides a good guide to the decision process within WLM.

One peculiar reservation about going to goal mode is a consequence of indiscriminate reporting of irrelevant data. If you've been reporting trivia like paging rates and CPU utilizations, and those who receive the reports watch those numbers, it might be hard to explain why you're not reporting them any more after going to goal mode. Recommendation: focus reporting on the attainment (or non-attainment) of service targets rather than on the details of resource utilization or activity rates—before beginning the transition to goal mode. (This data should still be available for capacity tracking and planning, but just not distributed to those who have no business need for such information.)

Recognizing that control of dispatching priority, MPL, and storage protection is indirect might lead some SRM control fanatics into an unfortunate attempt to actually control these values by warping the service policy. Such distortion might take the form of overly precise goals, *e.g.*, 1 second response time at the 87th percentile and 1.2 second response time at the 91st percentile for different service classes in the same policy. Tightly clustered velocities are another indication of an attempt to ask too much of WLM.

Another potential trouble opportunity is the desire to map an IPS to a policy without the kind of business analysis the task requires. While there might be a rough correspondence between performance groups and service classes, there is little that can carry over between the specifications in the IPS and the goals in a service policy.[16] It should be easier to go back to the analysis of needs, expectations, and business priorities than to attempt to divine what the goals might have been that led to the values in a particular IPS.

16 In OS/390 Release 3, there is a partial capitulation to the idea that performance groups can have a relationship to service classes. The value of the `PERFORM=` parameter of JCL is added to the set of classification rules.

Problems with WLM

Two kinds of problems have been reported by some early users. (These problems are not the same as bugs; presumably WLM is working as designed in these cases.)

The first of these is more properly a problem of an under-configured paging subsystem. The scenario is:

- An important transaction processing workload with a stringent response time goal has been running at peak capacity for some time. Because there is some central storage constraint, WLM has assisted the server address space(s) with storage protection for the current working set.
- The majority of the users of the subsystem become inactive for a protracted period of time. Perhaps there is a long "all hands" meeting—or everybody goes to lunch at the same time.
- While the users are inactive or there is a very long interval between transaction arrivals, the current working set declines, storage delay declines, and the need for storage protection passes. Unused frames are stolen, and (assuming there is no expanded storage or only a small amount of it) the pages end up on auxiliary storage.
- When the next transactions arrive, the working set of the server address space(s) is restored through a succession of page faults. Response time is unacceptably poor until the working set is reconstituted.

This "problem" endures because there is little or no expanded storage in the configuration and is exacerbated by an anemic auxiliary storage configuration. It can be avoided (or at least the effect can be minimized) if the paging subsystem were to be made more robust. The benefit of dynamic management of central storage allocation far outweighs the minor inconvenience of occasional delayed response time—if it is truly occasional. If the workload (assumed here to be CICS) has a very low transaction rate but business needs require consistently good response time, one possible solution might be to place that set of transactions in a pair of AORs on separate small MVS images, perhaps in LPARs, with a generous amount of central storage and *no* competing work. A simpler and more permanent solution might be for Workload Manager to correlate the rate of removal of storage protection inversely with importance.

The second problem is similar in some ways to the first. It too is a consequence of dynamic resource management replacing static management. However, it may be more nearly an intrinsic WLM behavior than a problem with some other subsystem. This scenario is:

- A batch service class with a low velocity goal, perhaps 15 percent, has several jobs running on third shift. Through contention with other work, the velocity goal for one job is not achieved until its dispatching priority is raised significantly.
- The service class "dries up" with the high-priority job as the last one to complete. WLM leaves the dispatching priority for the service class where it was.
- Some hours later, during first shift, another job comes into the empty service class. The job runs at the retained dispatching priority, now inappropriately high, and other work is impacted until the dispatching priorities are corrected by WLM.

As unlikely as this sequence of events might be, some do worry about it. One possible solution might be to run an unending synthetic job in the service class, one with a low demand for service, to ensure that the service class never goes empty, and that the required velocity is always achieved. However, it's more likely to believe that IBM may reconsider the wisdom of leaving dispatching priorities for empty service classes at their last level.

Limitations of WLM

In contrast with the common advice "If it ain't broke don't fix it," WLM can't fix what it don't know is broke. Network and JES delays (other than the input queue time) are completely outside its view, and ENQueue and most I/O delays are classified by WLM as "unknown." WLM has no control of the resources that might be managed to deal with such delays.

Most serious performance problems just happen to be in those areas. Managing CPU and storage delay just happens to have claimed a disproportionate share of tuning attention in the past because MVS managed these areas so badly. With Workload Manager in goal mode, these areas can now be left safely under the system's control, and performance management efforts can now be directed to the significant problems that formerly had to be pushed aside.

Therefore, performance measurement and monitoring might change emphasis but will remain just as necessary as before—and some of the real problems of applications may be solved.

Extrapolation

Although it might take some time before the role of WLM is broadened to take over comprehensive management of the initiators, the I/O subsystem, or the network, it might be possible for WLM to do some of the job relatively soon. It appears to be likely that WLM could soon become involved in managing ENQueue delay and adjusting the line-up of initiators within specified limits, based on policy and demand.

WLM can be expected to take on most mundane aspects of MVS performance management that can be readily measured and for which simple algorithmic methods can be devised.

The really good problems will be left for human beings to discover, characterize, analyze, and solve, with the assistance of ever-more-advanced performance and systems management software products.

7.6. Cheryl Watson's "Quickstart" Service Policy

Cheryl Watson, the president of Watson & Walker, Inc. has given permission for the recommendations of her "Quickstart" service policy to be included here.[17] The following tables are adapted from the summary charts in her paper.

7.6.1. Workloads, service classes, and goals

In the three tables that follow, the format, order, and some of the names have been changed from those in Ms. Watson's paper. Any errors thus introduced are the sole responsibility of the author.

Workload	Service Class	Period	Importance	Goal Type	Value	Percentile
	[SYSTEM][a]			system		

17 For further information, including how to obtain the full paper, please write to Cheryl Watson at:
Watson & Walker, Inc.
1605 Main Street, Suite 610
Sarasota, FL 34236 (USA)
or send e-mail to cwatson@netline.net, or call her at (800)553-4562.

Workload	Service Class	Period	Import-ance	Goal Type	Value	Per- centile
	[SYSSTC]			system		
	[SYSOTHER]			discretionary		
ONLINE[b]	ONLPROD		1	velocity	50	
ONLINE	ONLTEST		5	velocity	10	
STC	STCMED		3	velocity	30	
STC	STCLO			discretionary		
TSO	TSOPRD	1	2	response time	.5 s	80
		2	3	response time	2.0 s	80
		3	5	response time[c]	10.0 s	50
PRDBAT	PRDBATHI		2	velocity	30	
PRDBAT	PRDBATLO			discretionary		
ONLINE	OLPRODHI		1	response time	.5 s	80
ONLINE	OLPRODMD		2	response time	3.0 s	80
ONLINE	OLPRODLO		3	response time	10.0 s	50
TSTBAT	TSTBATHI		3	response time	10 m	90
TSTBAT	TSTBATMD		4	response time	30 m	80
TSTBAT	TSTBATLO			discretionary		
OMVS	OMVS	1	3	response time	1.0 s	80
		2	4	velocity	30	
ASCH	ASCH	1	3	response time	1.0 s	80
		2	4	velocity	30	

Notes:

a. The three bracketed service classes are system-defined.

b. The `ONLPRD` and `ONLTST` service classes contain the server address spaces; the `OLPRODHI`, `-MD`, and `-LO` classes contain the transactions of the WLM-enabled subsystems, CICS 4.1 or later, IMS 5.1 or later, or other similarly enabled subsystems.

c. For low completion rates, a velocity goal of 30 or 35% may be preferable.

7.6.2. Classification rules

Subsystem	Rule Type	Name or List Name	Service Class
TSO	default		TSOPRD
STC	TNG	STCHI	(omitted) for SYSSTC
STC	TNG	STCMD	STCMED

Subsystem	Rule Type	Name or List Name	Service Class
STC	TNG	ONLPRD	ONLPROD
STC	TNG	ONLTST	ONLTEST
STC	TN	*	STCLO
JES	default		PRDBATLO
JES	TCG	PRDBATHI	PRDBATHI
JES	TCG	TSTBATHI	TSTBATHI
JES	TCG	TSTBATMD	TSTBATMD
JES	TCG	TSTBATLO	TSTBATLO
ASCH	default		ASCH
OMVS	default		OMVS
CICS or IMS	default		OLPRODLO
CICS or IMS	TNG	OLTPHI	OLPRODHI
CICS or IMS	TNG	OLTPMD	OLPRODMD

7.6.3. Classification groups

Group Name	Type	Content
STCHI	TNG	VTAM, JESn,TSO, RMF, APPC,ASCH,OMVS kernel, performance monitors and automation packages
STCMD	TNG	schedulers, print subsystems, OMVS dæmons
ONLPRD	TNG	CICS*, IMS*, DB2*, other online system servers
ONLTST	TNG	test address spaces for online subsystems
PRDBATHI	TCG	production batch job classes
TSTBATHI	TCG	urgent non-production batch classes
TSTBATMD	TCG	important non-production batch classes
TSTBATLO	TCG	routine non-production batch classes
OLTPHI	TNG	high-importance, fast-response CICS or IMS transactions
OLTPMD	TNG	CICS or IMS transaction not qualifying for OLTPHI but with more than routine business importance

7.7. Summary

Workload Manager in goal mode brings a new level of automation to workload performance management in the MVS environment. Tasks that were difficult to perform at all in compatibility mode are now done behind the scenes by new extensions to the SRM. All it costs is a small increment of CPU time which is usually exceeded in magnitude by increased throughput from the hardware configuration. Complex directives in the form of SRM parameters are replaced by simple statements of the required level of workload performance and business importance.

These changes have the potential to relieve system programmers and performance management specialists of difficult and unrewarding tasks and free them to become as effective as possible in dealing with the problems of today and tomorrow. These include the performance and availability management of large heterogeneous enterprise configurations and networks, the availability and performance of the network, the performance problems of connectivity with the Internet and other server destinations, the unique problems of MVS as a large-scale server system, and the opportunity to improve the performance of key applications.

7.8. Chapter Questions

1. If you are already operating in goal mode, what do you expect to gain from it? If not, give three reasons (other than not being on Version 5 or later) for your choice. Analyze those reasons based on what you've learned in this chapter. Are they still defensible?
2. Name the five most important and the three least important segments of work on your MVS systems.
3. For the five most important workload segments, state the performance requirements that are justified by the business importance of each workload segment.
4. Are there any "generally accepted" performance requirements in your installation that are not justified by an economic or business reason?
5. Finally, create a service policy implementing the research of the three previous questions. If there turn out to be service classes defining distinct portions of a transaction-processing workload, indicate what changes may be needed in

the relationships between transactions and address spaces.

Chapter

8

Introduction to Performance Problems

8.1. Initial Example

Suppose that you are running a program to calculate π to 90,000 decimal places, and you need the results in one hour to gain an entry in *Grumbick's New! Book of World Computer Records.* On your fastest uniprocessor that can run MVS, the program requires 43 minutes of CPU time and will complete in about 49 minutes if it is the only nonsystem workload. However, under normal system loading conditions, the elapsed execution time is no better than 72 minutes.

The mission fails. There is an unacceptable mismatch among objectives, program, the computational problem, and the available resources. In such a case, "the problem" may be viewed in various ways:

- First, and most simple, the constraints may be too stringent. Often a rough estimate becomes a firm requirement simply because no refinement to the estimate was available before the project gained momentum. There might have been an excessive safety factor in this part of a problem definition—perhaps the record to be broken is only 60,000 decimal places in an hour and a half.
- Scheduling may be inappropriate. The job may be too CPU-intensive to be suitable to run on prime shift. At 3 a.m., the job may complete well within the target.

- Contention may be responsible. The elongation of elapsed time might be due to contention with other work, even off-shift. If the job is given a more favorable goal, perhaps with a higher velocity, other work in the system will suffer, but there is some chance that the target elapsed time will be achieved.

- The program may be poorly designed or implemented. If the numerical approach embodied in the program is amenable to multitasking or to vectorization, adding to the CPU resource (with a multi-engine processor complex or a vector processor, or both) may be effective. Some of today's compilers can generate programs that take advantage of multiprocessors or vector processing features. By exploiting multitasking or vectorizable content, the impossible might become easy.

  Batch jobs more conventional than the example might be good candidates for parallel splitting. The job might be split into a suite of jobs or, using the facilities of IBM's SmartBatch or similar products, into a set of pseudo-jobs that still retain their original structure. Besides making it possible to execute steps in parallel, this approach makes it possible to use the BatchPipes facility to eliminate the creation of intermediate data sets.

- The fundamental algorithm or design embodied in the program may be flawed. The greatest improvement in execution time is often associated with rethinking the fundamental design of an application program. In the case of a compute-intensive problem like the one we're considering, even a small change in the numerical algorithm might cause the calculation to converge much faster and thus have a profound effect on requirements for computing resources.

8.2. Types of Performance Problems

Even though the foregoing was an oversimplified and contrived example, note one key element: The way we recognized a "performance problem" was to note that **a workload failed to achieve its service goal.** Without that distinguishing factor, we tend to fall into a vague search for improvement, without

knowing when to stop. Such a lack of focus makes for frustrated performance analysts and a system unresponsive to legitimate performance problems. Establishing performance targets and assessing actual performance against those targets provides a rational basis for performance management activities. Assessing measurements can take place in real time for interactive workloads and key unattended workloads, or after the fact for ordinary batch workloads. Performance monitors capable of making such measurements and more will be discussed in Chapter 9.

We now examine, in a more abstract way, the kinds of problems we've touched on so far:

- **Unrealistically ambitious service expectations.** To give a common example, attempting to achieve 1.5-second CICS response time at the 90th percentile may be unreasonable if most transactions are more than ordinarily complex. A typical user of the application's transactions may require 30 or 40 seconds of think time and data-entry time to fill a complex screen. Response time is thus a minor component of task service time. Sizing a system to provide requested but unjustified performance at initial activation of an application system may be a waste of money if the response time turns out to be 50 percent better than what is needed to meet the service requirement of the using department.

  Attainment against properly planned and structured service targets may be measured, and exceptions may be made visible in real time or after the fact. What is to be done with such information? One approach is to simply report the results, taking action to correct "anomalies" only when the delivered service in a reporting period has become overtly unacceptable.

  The approach we characterize in these pages as **performance management** is more activist. If those in charge of providing the service have a level of self-measurement more stringent than the commitment to the end-users of the service, response time anomalies can be detected before the customers notice them. If, in addition, the service providers have diagnostic and repair tools to allow corrections to be made in real time, there need be very few reportable

service deviations. Goal mode with Workload Manager provides an opportunity to set goals that represent a bit of overachievement, so that when a "real" target is missed WLM will already be trying to improve the performance of that workload.

- **Poor scheduling.** Often an installation has no service target for unscheduled batch and no incentives (or disincentives) to condition the expectations or behavior of the users bringing batch jobs into the system. Consequently, batch jobs varying wildly in resource requirements, setup need, contention potential, and business value compete freely. If some scheduling rules or preferences (including incentives to avoid contention and substantial financial or budget incentives to wait longer for results) were to be made known to the submitters of the jobs, enough people might alter their behavior to result in a substantial reduction of resource utilization on prime shift. Moving "heavy" batch off-shift can help to hold off a planned upgrade in capacity.

- **Avoidable contention.** Although end-users or application owners can rearrange work to avoid contention that they are warned of in advance, most resource contention is not visible until an unanticipated service level problem appears. Resolving such problems involves many paths and is often a principal activity of those responsible for performance management.

- **Poor implementation or installation choices.** An application program may have an inspired design and be written flawlessly. It may then be installed for production by someone who is careless or uninformed about the capabilities and services to be found in an MVS environment. As simple a mistake as placing a large card-image file in an unblocked sequential data set may create avoidable stress on a device, a string of devices, a storage director, a channel, and one or more CPUs. Because program residency takes longer than it should, real storage may be excessively utilized as well.

  In the OS/390 or MVS/ESA environment, failure to understand and make effective use of BatchPipes or its successor SmartBatch, shared pages, data spaces, hiperspaces, and Data in Virtual, as well as the more prosaic VIO and buff-

ering options available since the beginning of MVS, can result in inefficient and wasteful use of system resources.

- **Flawed basic design or application architecture.** Even though poor follow-through in installation can lead to inefficiency, it takes an inherently flawed design to make a program spectacularly deficient. Stories abound of inept program designs. A typical pattern is that an application program survives long beyond its originally expected life span, and that the conditions of its execution change enough to make it run poorly.

  Sometimes an increase in volume alone is enough. Sorting algorithms, in particular, are very sensitive to volume. A correct choice for sorting 100 items may be hopelessly inefficient for 100,000. However, the high-volume sorting algorithm may require the execution of so much setup code that its overhead is unacceptable for a frequently invoked in-line sort of a few hundred items. A "one size fits all" optimal sort is very unlikely.

  A scale-up problem should be suspected when a previously acceptable application program starts missing its service targets chronically, especially when a gradual increase in its input volume has occurred. Confirmation may be gained by doing a "reasonableness check" on the gross activity of the suspect program. Is the increase in run time proportionally greater than the increase in input volume and output volume? Is much more intermediate I/O done with only a small increase in volume? Does CPU consumption or real storage requirement jump, again more than proportionally to the volume increase?

  In such cases, time devoted to digging up long-forgotten source code and reanalyzing the fit of the application to today's hardware may be well spent.

8.2.1. Capacity planning and performance management

When workloads and consequent hardware resource use grow over time, the management of such concerns is usually considered to be purely a matter for capacity planners. However, if the design of a principal application is of the kind just described, the hardware may be slated for an expensive upgrade before it is really necessary. More-than-proportional increases in run times of

key applications should cause those applications to be reexamined as part of the upgrade justification process.

8.3. Resource Contention

To a great extent, performance problems originating in a lack of capacity are simple to solve. Add more capacity, or improve the underlying program to relieve the lack of capacity, and the problem is gone, or at least postponed.

Much more challenging problems are possible. They occur when multiple workloads need concurrent access to specific system resources. The delay in transaction completion caused by resource shortage might be a form of simple elongation: If 40 minutes of CPU time is required to complete a job that required 20 minutes before a volume increase, the job's run time will extend by little more than 20 minutes.

If, however, there is contention with other workloads for the CPU resource, the additional queueing delay may be many times the elongation due to additional resource use. If the resource in contention is DASD, an even greater delay may be introduced in all contending workloads, depending on the amount of actuator movement (seeking) added to the I/O time of each.

If a resource contention problem is known to exist, the measures needed to deal with it are different from those used to diagnose and solve resource exhaustion problems.

8.3.1. Workload assignment

In a multisystem environment, inappropriate workload assignment choices may lead to avoidable contention. For example, in systems with shared DASD, in which GRS or an equivalent is not managing cross-system ENQueues, an exclusive ENQ request causes a RESERVE to be placed on the volume containing the data set. If other data sets on the volume are required by a system other than the one from which the RESERVE was issued, access is blocked for the duration of the RESERVE. This kind of problem should vanish as parallel sysplex comes into common use.

Such denial of access can cause long delays if an interactive subsystem or set of users needs an affected data set. The original reason for creating the potential contention might have been an attempt to separate TSO users maintaining an IMS subsystem from the production system providing the IMS service. If any

data set used by a TSO user is on a volume needed for production, delays caused by RESERVEs are possible.

Even if GRS or an equivalent facility is used to eliminate most RESERVEs, activity on the "off-side" system(s) can cause delays on the production system. To get a complete picture of shared DASD contention requires merging usage data from all sharing systems. Predictive modeling is badly flawed if such a complete picture is not input to the model.

As has been the case for many kinds of problems in the past, this one too may be solved eventually. If the systems in contention are in a parallel sysplex, the type of data set may determine whether there is going to be contention delay. Access methods that are enabled for data sharing will hold the scope of contention down to the lowest possible level—typically a single record.

8.3.2. Workload scheduling

Sometimes workloads damage each other so much through contention that they cannot coexist. A daily transaction analysis and validation batch job operating on an online system's data base might cause the devices serving the application to become so busy that acceptable response times are not possible. Daily backups of critical disk volumes are in the same category. An efficient utility program performing data backups will drive its devices at nearly 100 percent utilization. Something must be done to keep these necessary batch jobs from destroying the performance of the interactive application. What alternatives are available?

- If there is a daily, naturally inactive period of sufficient duration to accommodate the daily jobs with enough spare time to handle at least one rerun of the longest-running job, a simple scheduling approach may do. Such an interval is usually known as the "batch window." Performance management action, including real-time monitoring of production batch completion times, and close capacity-planning tracking are needed to ensure that the time constraints are *always* safely met.

- Many online applications must be available virtually all the time. Less antagonistic and less resource-intensive approaches to backup and auxiliary batch operations are necessary. Volume backups might be done on a weekly basis instead of daily, supplemented by incremental daily backups of only changed data sets. New DASD subsystem capa-

bilities such as Concurrent Copy and Remote Copy ease the pain by not tying up system resources to make point-in-time copies of data sets or volumes. Validation and adjustment might be done against transaction logs, with an exception set of amending transactions being generated for entry through a less intensive path than batch. A personal computer might be used in terminal-emulation mode to drive simple transactions at a relaxed rate.

- As the use of RAID devices takes hold, the need for backups may change to be more oriented to the business cycle and less to the fear of unrecoverable data loss.
- Eventually, applications might well need to be designed for continuous operation, generating parallel data streams for backup, and performing validation and amendment in concurrent processes. Today's large multiprocessing systems are capable of dealing with such applications easily, although the software is not quite as ready as the hardware.

8.3.3. Accommodating contention

One of the problems pointed out earlier is that without service targets, performance management has no endpoint. Approaches that focus on data about resources and infer the possibility of contention for those resources might lead to numerous "corrective" actions to reduce service time, utilization, queue length, or the appearance of contention. Without input from a workload-oriented point of view, such adjustments may be unnecessary at best. A balanced viewpoint is essential. Starting with the workload view, the prime justification for a "tuning" or "resource-balancing" action would be a failure to meet a service target, or a clear trend indicating that such a failure is imminent. This is the approach at the center of Workload Manager.

Based on such evidence, an investigation of specific resources tied to the threatened service level can lead to the solution of a real problem.

8.4. Special Problems of Client-server Applications

Chapter 11 covers performance considerations for client-server applications in some detail. Client-server is becoming a mundane fact of life today and IBM has taken to referring to the

OS/390 hardware platforms as "servers" or "enterprise servers." Client-server can show some unique problems. Just to cite one example, consider the splitting of former "dumb terminal" applications to put the control of the user interface on the client system, typically a PC work station.

The problem goes something like this: suppose that the former user interface code is not neatly isolated, or that there are numerous small functional routines that constitute the user interface implementation. In the predecessor application, the flow of control between the mainline code and the user interface code, through standard program linkages, is innocuous. There's no problem if it takes 300 *CALL*s to build a screen.

Now consider what happens when those 300 *CALL*s become DCE Remote Procedure Calls (*RPC*s). Each instance of the linkage now represents a line turnaround, an RPC directory search, and all the necessary protocol stacking and destacking, and character set translations, as well as the basic interceptor code that "catches" the *CALL* and turns it into an *RPC*, and the corresponding code that receives the *RPC* and presents it to the called program as if it were an ordinary *CALL*.

If the user interface protocols were simplified and lifted to a higher level, perhaps the 300 *CALL*s could be reduced to a dozen or fewer. With no change in the function of the client-server application, service could be improved, and resources could be saved.

Other considerations for such applications may be found in Chapter 11.

8.5. Summary

In this chapter, we have attempted to identify several typical ways in which workloads can fail to meet their service targets. We've also suggested ways to monitor the workloads and measure their performance, as well as performance factors related to system resources. The discussion is by no means complete, and the "war stories" of how performance problems manifest themselves have no limit. In the next chapter, we'll examine measurement and monitoring methods and tools.

8.6. Chapter Questions

1. Investigate the performance problems in your system. Classify them according to the types discussed in this chapter. Does the exercise of classifying the problems contribute to your understanding of them? Does it suggest solutions?
2. Does your installation have service level agreements? How closely do your reported results match what is required? Do you expect that user satisfaction will erode if performance declines but still stays within the committed levels?
3. How do your site's DASD management policies relate to problems of contention? Are volumes owned by the data center or by the using department? If the latter is the case, why is it so?

Chapter

9

Measurement and Prediction

Any performance management action is initiated based on a decision—by someone—that action is necessary. The decision is in turn based on objective information placed in a subjective context appropriate to the needs of the business:

- Is a customer complaining of poor service?
- Has any condition occurred in the last 24 hours that requires corrective action?
- CICS response time is not acceptable for users in Accounts Receivable. How extensive is the problem and what is the appropriate action to take?
- Has a short-term tuning action introduced a new problem?
- Can this system accommodate 50 more TSO users?
- What is the most effective way to spend my hardware upgrade budget this year?

All of these determinations require information in order to be made effectively and efficiently. There are two basic sources of such information: measurement and modeling.

Measurement takes place at many levels within MVS and is concerned with a wide variety of data. *Resource* data may be collected to learn the extent of CPU, channel, and device utilization over various time intervals. *Activity* rates in the system are

gathered, analyzed, and displayed or stored. *Workload* data of many varieties may be measured in various ways. In MVS since Version 5, the Workload Manager collects much of the data that was formerly collected only by external measurement products. Vendors of those products, in turn, have taken the opportunity to reduce their impact on system performance by using the WLM measurement data made available through an IBM-supported Application Programmer Interface (API), supplementing it with data that they collect for analyses beyond WLM's scope.

Measurement data will suffice to answer most questions and provide support for most day-to-day decisions in performance management. When the question concerns future expectations or seeks to understand processes for which measurement data is impossible to obtain (or just not available with the current set of tools), *modeling* may be the appropriate discipline to use.

Modeling falls into two major categories, similar in spirit to the principal schools of stock market analysis. *Analytic modeling*, through an understanding of the way in which a system works, seeks to create a mathematical model whose behavior mimics that of the system. Analytic modeling resembles the "fundamental" approach to market analysis. *Stochastic modeling*, like "technical" stock market analysis, looks at behavior patterns and examines and projects those patterns, largely independent of internal considerations.

We now examine various approaches to performance management and to measurement, summarize the kinds of decisions aided by each kind of measurement, and then illustrate one aspect of analytic modeling technique by developing a piecewise analytic model of DASD I/O. However, a full discussion of analytic modeling, as well as any consideration of stochastic modeling, is beyond the intended scope of this book.

9.1. Approaches to Performance Management

Differences of approach are common in performance management. They derive from differences among tools and the vendors who supply most of the tools, as well as the background and experience of individual practitioners.

The way in which data processing relates to the overall business has an effect as well. Data processing may be relatively new in an establishment and its advocates untrusted by senior manag-

ers. In such an environment, capacity planners must constantly struggle to justify both today's DP budget and the essential relationship between business growth and DP capacity growth. They must also work to persuade a skeptical or hostile audience that growing dependency on DP leads to ever more stringent expectations for performance and availability, and a corresponding growth in the need for performance management tools.

In such an environment, tracking and accounting for resource usage is a daily concern, so it is natural that performance management is approached from the perspective of resource use and its cost distribution. The early history of performance management took place in a time when the cost of hardware per transaction was much greater than it is today, so the resource-oriented point of view is well entrenched for historically valid reasons.

9.1.1. Resource-oriented performance management

When looking at resources, how is a performance problem identified? When a pattern of satisfactory operation has been established and key measurements taken, *any* deviation from history is suspect and might signify an abnormal condition. For example, if the CPU is usually about 75 percent busy at 2:00 p.m., for utilization to drop suddenly to 50 percent might mean almost anything—but not that the CPU has instantly become more powerful! If such a deviation from normal operation is detected in real time, further data may be gathered and possible causes and consequences might be determined, depending on the tools available.

The implicit assumption of resource-oriented methods is, "If the numbers look good, the system looks good." Which numbers? Ideally, the critical numbers should be determined by each installation according to its own measurements. What often happens, however, is that less experienced system programmers and performance analysts seek out help from others, requesting advice in the form of questions like:

- How busy should the CPU be?
- What's a good paging rate for a ES/9000-962?
- How high can device and path utilization go on a DB2 data base on RAMAC 3?

As the numbers become detached from the experience and expectations for which they were appropriate, they degenerate

into mere "rules of thumb." At this point, a system might well be measured against someone else's operating limits, and for a different kind of workload. Depending on such methodology often produces results that reinforce the atmosphere of distrust touched on above.

The resource-oriented approach is inadequate because it promotes a focus on measures of resource activity rather than on the service achieved on behalf of key workloads. Resource data in the Workload Manager era now reverts to the purpose for which it is far better suited—tracking and accounting for the use of system resources.

9.1.2. Workload-oriented performance management

Even though inferences about workload performance factors can be drawn from resource measurements, it is difficult. A performance management methodology built on resource measurements is uncertain and often too dependent on archived resource usage data to be useful in solving an acute problem. A more direct approach is necessary and is made possible by performance monitors that view the resources of the system from the perspective of its workloads, or the workloads from each other's viewpoint. Workload Manager does a comprehensive job of collecting state data for CPU and storage, but it does not collect state data for ENQueue and network delays, nor does it collect contention data in any direct sense.

As of OS/390 Release 3, Workload Manager collects I/O component data, consisting of the four customary components of I/O: IOSQ, PEND, CONN, and DISC. This data is collected at the device level but is immediately aggregated to address spaces. Consequently, device-related analysis requires independent data collection—unless IBM, perhaps in a later OS/390 release, makes the raw data (before aggregation) available at an API.

Most measurement and management tools operate by sampling the execution states of the workload(s) of interest. For contention analysis, samples that suggest contention delays may be pursued further, to discover the identity of the current resource holder. The accumulation of sampled data provides a pattern describing how often the workload is found in each state. We look now at how such information can help to solve problems.

Execution-State Analysis

To determine how to deal with a missed service target, a straightforward problem-solving approach will attempt first to identify the problem's cause. Typical causes include:

- A hardware component may be degraded or operating poorly. A performance-enhancing feature such as a high-speed buffer might be disabled, a CPU or some storage could be offline, or a path to a high-use I/O device might be inoperative.
- Available resources are insufficient to permit the service level to be met. Application data volume might have increased suddenly or over time, causing greater resource need.
- Changes in the application program, the system environment, or the amount or distribution of stored data might have introduced inefficiency in the way the application executes. The introduction of extra SEEKs into frequently executed uncached I/O operations (as in the case of a randomly accessed data set that grows into widely separated multiple extents) might cause unacceptable increases in run times or transaction response times.
- Growing demand for nonspecific system resources such as storage or CPU cycles could cause delays while waiting to gain access to sufficient resources to complete a work unit.
- Another workload might be using a specific resource (usually an I/O device) needed by the workload that is failing to meet its service target. Contention for an I/O device will elongate the elapsed time for all competing address spaces. If a data set is needed by both the Loved One and the source of contention, and the contention source has exclusive access, the affected workload will receive no service until the ENQueue bottleneck is broken.

The first of these possible causes might be diagnosed by a very complete resource availability monitor. The second might be spotted after the fact by analysis of SMF or RMF resource data. The others are virtually impossible to diagnose without execution-state analysis or direct contention analysis, or both.

In a workload-oriented methodology, the focus is first on creating service targets, then on measuring their attainment, and fi-

nally on taking corrective action when targets are missed. The measurement and management actions can be done directly and invisibly by Workload Manager, or, when goal mode is not active, through the use of suitable performance monitoring tools and a logical tuning approach.

The key emphasis in a workload-oriented methodology is on *service* measurement. Once a deviation from a service target is identified, investigation can proceed with resource-based tools, or with more modern tools based on the decomposition of transaction times into their execution-state components.

Using resource-oriented methods to diagnose problems is indirect and uncertain, but a product opportunity has materialized to take advantage of these very characteristics. Several vendors offer "expert systems" to evaluate resource data and "diagnose" whatever problems might be represented in the data.

If resource data were all that was available, this approach might be the best possible. Consider the massive investment in early supercomputers used for weather forecasting, particularly for predicting storm tracks. A mass of temperature, pressure, and wind velocity readings rimming the Gulf of Mexico and the Caribbean might have been combined to give an imprecise prediction of a hurricane's time and location of landfall. The programming of such computers represented the "expert systems" of their day.

Of course, all we need to do today is use the detailed observations from weather satellites to extrapolate the directly observed storm track and velocity. The availability of direct observation makes the tedious "expert" analysis of indirect data unnecessary. Likewise, workload-oriented performance measurement tools make it possible to diagnose workload problems directly and rapidly.

The most successful example of the workload-oriented technique, carried through from measurement to management of workload performance to goals, is Workload Manager in goal mode.

9.2. Classifying Measurement Tools

There are several ways to classify performance measurement products. One frequently used division is between *real-time monitors*, which present data as they collect it, and *data collectors* or historical monitors, which store data for future use. In

some measurement products, both approaches are taken; some data is immediately available, while data representing past conditions is stored.

9.2.1. Real-time monitors vs. data collectors

How does one choose? While it is certainly true that a creeping increase in response time may be diagnosed by a series of real-time measurements, why bother? A data collector with a workload-problem-oriented reporting program can present the history of the problem in summary or graphical form, guiding the analyst to a solution that is not unduly influenced by instantaneous perturbations.

On the other hand, a problem such as a sudden increase in allocation of the Common Service Area (CSA) must be revealed and understood promptly. Action must be taken to find the cause and stabilize CSA use before the system "crashes" when a crucial function cannot obtain needed CSA. A real-time monitor with a constantly active "alert" capability provides the best way to become aware of such a problem. Because simple awareness is not enough, a more complete real-time monitor would provide additional functions to identify the holder of CSA accounting for the most allocation or for the most rapid rate of allocation growth,[1] as well as some means of freeing a CSA block in an emergency.

A real-time monitor tends to be primarily oriented to basic concerns affecting the availability of the hardware and software configuration. It is not *per se* a performance monitor. The exhaustion of CSA is an availability problem, not a performance problem. Typically, a real-time *performance* monitor will also evaluate service data and emit an alert if a critical response-time or elapsed-time service level is in jeopardy. It may also include a facility to decompose those times into their component parts, so that the most likely cause of a service anomaly may be quickly identified.

As capable as real-time monitors may be, they are not the best choice for the more deliberative job of correcting subsystem or application performance problems. Such service anomalies tend to be chronic or cyclical. An anomaly does not arrive on the scene as a full-fledged crisis some fine day, and it does not

1 Even in a "real-time" monitor, the data to support these functions must be gathered before it is needed for display. The essence of the real-time monitor is in the immediacy and current relevance of data presentation, not in the manner or timing of its collection.

politely stay visible until someone is looking at the display of a real-time monitor.

A data collector has advantages as a measurement tool for performance management. Only the collector portion need run at high dispatching priority, to ensure accurate CPU-delay measurements. (In MVS as of Version 5, even this intrusion on useful work is unnecessary. WLM collects state data continuously.) The data analysis and display portion can run as a batch job or as a TSO command or CALLed program, or as an independent APPC/MVS transaction processor. A client-server design could allow the reporter to reside on a different platform from the MVS system being monitored. If the data is stored online in a form the reporter can query directly, the reporter's immediacy can be close to that of a real-time product.

If instead the standard MVS System Management Facility (SMF) is used as the data repository, there can be less overhead attributable to the collector.[2] However, reporting must be deferred until the SMF data is dumped and processed. With the availability of the System Logger in SP 5.2.2 and OS/390, SMF processing and access to the data may entail less delay.

Another facility, the Sysplex Data Server, is part of RMF since Version 5.1. The Sysplex Data Server can coordinate RMF data from all systems in a sysplex, and includes a set of services to permit other products or programs to have similar access to all types of SMF data. The data provided in the resulting RMF displays is not long-term historical data but the more instantaneous data characteristic of Monitor II.

There are other benefits of routing performance measurement data through SMF. Writing out the data is automatic, through facilities provided with the operating system and designed for continuous operation. There are also several complementary products available for data analysis and presentation. These include IBM's Service Level Reporter (SLR) and its successor Enterprise Performance Data Manager/MVS (EPDM), and SAS Institute's Statistical Analysis System (SAS®), as well as higher-level data base and reporting packages built on SAS. (These include Computer Associates' MICS® and Merrill Consultants' MXG.®)

2 The data is placed in the SMF buffers and system routines write it out. The resources used for this purpose may or may not be "captured," but they will certainly not be associated with the data collector.

9.2.2. Measurement techniques

Another basis for classifying measurement products is according to whether data is obtained by sampling, by direct access to existing summary information, or by intercepting events. All of these methods are commonly used. IBM's Resource Measurement Facility (RMF) is primarily a sampling system. In MVS/370, RMF obtained data about I/O devices by sampling the states of devices as reflected in data areas of the operating system. In MVS/XA and MVS/ESA, however, the channel subsystem includes architecturally defined measurement interfaces, so what once had to be sampled can now be "harvested" using supported interfaces. In Candle Corporation's OMEGAMON II for MVS for example, the I/O data is taken from the channel measurement blocks, execution-state data for address spaces is sampled to the extent that WLM data needs augmentation, and CPU usage data is based on the accumulators updated by MVS at every dispatcher transition and on WLM's low-overhead collection.

Transaction-oriented measurement systems attempt to capture data relating to each transaction of interest. To do this, they are event-driven, capturing and buffering the key data for each transaction and moving it to a logging facility as quickly as possible. Data composition, reduction, and analysis are done at a later time.

MVS includes a "hidden" set of data collection facilities most commonly used by the Generalized Trace Facility (GTF). Distributed throughout MVS system code are numerous implanted MONITOR CALL (MC) instructions. Each such instruction has within it a 4-bit monitor class field. Normally, the MC is a "no-op." An authorized program may load appropriate control register settings to activate the MONITOR EVENT facility and designate active monitor classes. Then an MC designating one of the selected classes, when executed, causes a program check. Control is then routed through a specialized second-level interrupt handler to code that will eventually log some data associated with the monitored event.

MONITOR CALL is a disruptive instruction to use. Each time the program check occurs, the system undergoes the flow-of-control disruption of a status switch and spends a portion of the subsequent processing time disabled for all inter-

rupts on all processors of the complex. The processing time for the interrupt is tacked on to the TCB or SRB time of the program incurring the interrupt.

Consequently, the use of GTF is considered abnormal; most installations require special authorization to allow its use, and then only for limited time intervals. Granted, GTF is very accurate and can collect data that is available no other way; it is simply incompatible with production operation of an MVS system, and its high overhead causes inflation of reported CPU times throughout the system. A performance monitoring program gathering data by means of MONITOR CALL obviously should not operate continuously in a production system. Selective and limited use of MONITOR CALL can be effective if the event-related data it gathers is useful and available no other way.

Depending on what information is required, composites of these basic techniques are common. For instance, RMF accumulates TSO response-time data through each collection interval by event-driven capture of transaction completions but discards the individual transaction data. In RMF, the individual transaction data is lost by the accumulation process. Candle Corporation's OMEGAMON II for MVS can "reach over" into RMF's accumulators and display quasi-current values for TSO response time. Legent's (now CA's) TSOMON starts with the same basic transaction completion data as RMF, but by capturing it without aggregation, TSOMON can later display much more detailed information about TSO response time by user subset or by transaction identity. (The need for such information diminishes with the response time distribution information available directly from WLM data.)

9.2.3. Resource measurement

We'll look briefly at IBM offerings for resource measurement, followed by a look at those offered by other vendors.

IBM Offerings

Resource Measurement Facility (RMF). Although basic measurement data at the workload level is available from SMF, it is usually necessary to have an overview of how the system is performing as well. Understanding the relationship between system capacity and workload is made possible using data collected by RMF and written to SMF data sets in the form of records in

the type range of 70 through 79. RMF evolved from a no-charge facility, MF/1, that was part of SVS and early MVS. RMF reports on CPU, channel, and device utilizations; system activity rates; and the distribution of service by performance group or domain (in compatibility mode) and by service class and report class in goal mode.

Using RMF data, estimates of workload delay factors may be inferred as well. Delays for swapping, IOS queuing, paging, and CPU queuing may be calculated from RMF data. Information to quantify these delays is not provided directly in RMF reports, but rather derived through mathematical and statistical manipulation according to implicit or explicit models of system and workload behavior.

Independent Vendor Offerings

A number of independent software vendors produce resource-oriented performance monitors. They include Candle's OMEGAMON II for MVS, Boole & Babbage's Resolve Plus, Computer Associates' CA-LOOK, and Landmark's The Monitor (TMON) for MVS. These products also include features to monitor hardware availability and configuration elements, as well as some measures of service, delay, and contention. The historical component of Candle's OMEGAMON II for MVS (and each of its counterparts in other vendors' offerings) draws on both resource data (in turn "harvesting" data from RMF) and execution-state data as described below.

Value is added by such products. Landmark's TMON for MVS and Candle's OMEGAMON II for MVS provide views of many performance-related values in MVS, arranged in task-oriented screens with advanced navigation capabilities, going well beyond the rudimentary displays of RMF.

For example, RMF Monitor II provides high UIC numbers for the system and for address spaces. In systems with substantial storage isolation, such numbers provide a misleading view of page age and no information about reference patterns. The third-party offerings provide more detailed views of the page ages in address spaces and in the system, by age category and even by specific UIC values, perhaps going so far as to show UIC per page frame as well as an overall pattern of page age for all the virtual pages in the address space. Such information can help to detect incorrect storage isolation settings and to identify

storage reference "hot spots" when tuning an address space to improve efficiency or minimize system impact.

9.2.4. Workload-oriented measurement

Workload performance problems are most effectively diagnosed using tools that examine the workloads and expose contention, bottlenecks, and tuning opportunities.

Execution-State Analysis

The basic technique of workload-oriented measurement is execution-state analysis. By examining the state of each address space of interest at regular intervals (sampling), a frequency distribution of execution states may be constructed, as shown in simplified form in Figure 9-1.

The ranking of execution states can be used to characterize the problems (if any) associated with the workload. If the address space of interest is not meeting its service target and "using CPU" is the dominant state, the appropriate "cure" might be a more powerful CPU or a more efficient program.

Dominant states like "waiting for CPU," "swapped out for MPL," or "waiting for paging" represent resource shortages as seen by the address space. Such delays might be viewed as contention, but the workload in contention would be identified as "anything else." In such a case, the classical tuning tradeoffs are exercised: If other, less important work is meeting its service goals (or has

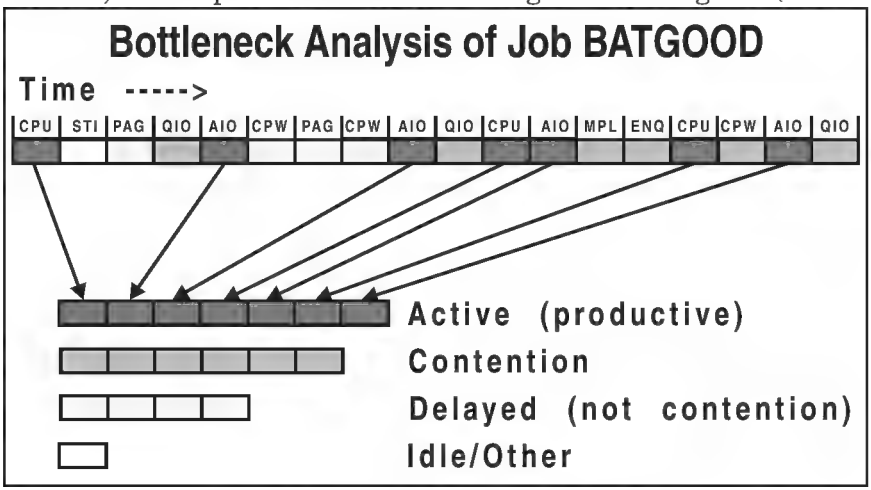


Figure 9-1. Simplified Bottleneck Analysis.

none), system parameter changes (or the Workload Manager in goal mode) can adjust dispatching priority, domain constraints or contention index, or quantity of storage isolation, thus redistributing resources. If such tradeoffs are not available, a hardware upgrade might be needed.

If the Loved One spends much of its time waiting for an I/O device or for an ENQueue resource, a contention problem with a specific workload is a likely cause. More investigation is needed to see if the contention arises from within the critical workload, or if it is caused by another address space or the aggregate of many address spaces, such as TSO users.

IBM Offerings. IBM's Resource Measurement Facility (RMF) includes an independent set of functions known as Monitor III. Another name for Monitor III is the *Workload Delay Monitor*, the name of an earlier IBM Research Division internal tool, ideas from which were included in the design for Monitor III. Monitor III depends on a separate data collector address space called RMFGAT. A reporter called RMFWDM is invoked as a TSO command and is used to display the data collected by RMFGAT, both in real time and with some capability to study past periods.

Monitor III supports a concept called *workflow,* defined as "using" (a resource or set of resources) divided by the sum of "using" and "delayed" (for the same resource or aggregate). If a TSO workload is idle 80 percent of the time, absorbing CPU cycles 5 percent of the time, and waiting for some resource 15 percent of the time, the workflow of that workload is $\frac{5}{5+15}$ or 25 percent. Note the resemblance between the notion of workflow to that of velocity as used in goal mode. The differences are that velocity can include in its definition delays not necessarily related to the resource being used, and that workflow is usually viewed for a single type of resource at a time.

Workflow is applied to both workload and resources in Monitor III, and the methodology suggested by the product documentation favors high workflow. Since the epitome of high workflow is a CPU loop, this view of the world is not accurate for all cases.

In addition to workflow, Monitor III shows how a workload's time is spent. Time is broken into processor use, processor delay, storage delay, and I/O delay. Other states are lumped as "other" or "unknown." Finer breakdowns are available for I/O delay, identifying the devices causing the most delay to a workload.

This brief description of Monitor III is necessarily incomplete. The appropriate IBM publications provide full information.

Independent Vendor Offerings. Candle Corporation offers a technology called variously *degradation analysis*, *bottleneck analysis*, and *execution-state analysis* in OMEGAMON II for MVS. The same techniques are used in other Candle products supporting the IMS and CICS subsystems. Data is collected by sampling the states of address spaces and supplemented by the CPU time accumulation kept in MVS data areas. The accumulated data is presented in real time, and collected for historical analysis as well. OMEGAMON II for MVS can also write its historical data to SMF, either alone or in parallel with writing to its proprietary data store.

Candle's analysis is somewhat more complete than that of RMF Monitor III, reporting on time spent using or waiting for each I/O device, on each specific ENQueue delay, and on numerous other execution states.

Other vendors, including Landmark and Boole & Babbage, also provide some form of execution-state analysis in their MVS monitoring products.

Direct Contention Analysis

When a favored workload is delayed because another workload is using a needed resource, knowing the details of execution states is not enough to unclog the bottleneck. Using resource data to supplement execution-state data helps a bit but can be misleading. For instance, if CICSA has spent 35 percent of its active time in the past 15 minutes waiting for volume DATA77 on device 03B7, it is naive and ineffective to simply consult an RMF Monitor III report to see who the other users of the device are.

The users accounting for the most time may not be causing the interference; it may come from CICSA itself. It may come from another address space using the device synchronously with CICSA, because it is driven by service requests from CICSA. A very small percentage of device utilization may account for a large measure of contention. While it is almost certainly true that one of the listed users of the device is the culprit, the ranked list does not always guide the analyst to the true source of contention.

What is needed in this case is a different kind of tool. A sampling execution-state monitor can be extended to perform a deeper kind of analysis when delay states are found. If such a monitor finds queued I/O delay for CICSA, the monitor goes beyond simply identifying and counting the delay state. A contention analyzer will examine other information available in the system to determine which address space is using the device in contention *at the time the delayed address space needs it.* By accumulating counts for each such occurrence, the monitor can produce a ranked listing of contending address spaces. A simplified view of direct contention analysis is presented in Figure 9-2.

IBM Offerings. RMF Monitor III collects true contention information for CPU contention only. For other resources in contention, Monitor III shows what might be termed *coincident usage* data. In the case of DASD contention, Monitor III shows a ranking of the address spaces using a device in contention without determining whether their use was in fact in contention with the workload of interest.

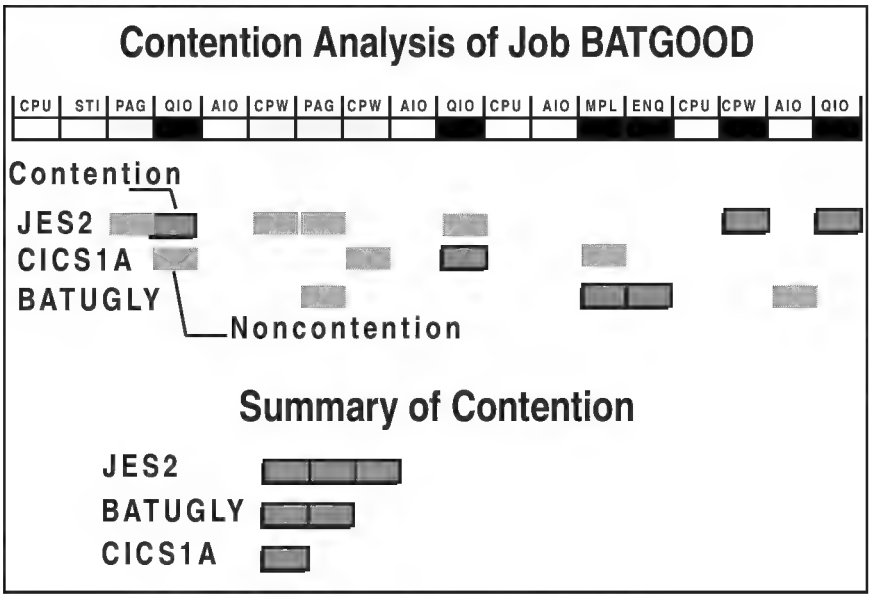


Figure 9-2. Direct Contention Analysis.

Independent Vendor Offerings. Candle Corporation's OMEGAMON II for MVS includes a true contention analysis capability known as Impact Analysis. Impact Analysis, first made available in Candle's OMEGAMON for CICS, can produce ranked reports of contention against single address spaces, performance groups, or arbitrary workloads. The contention is attributed to address spaces or performance groups. The latter can be broken down to address spaces within performance groups. The resources in contention can be displayed as well, with optional detail down to device numbers (with volume serial numbers) or individual ENQueue names.

9.2.5. Change monitoring

Large computer systems are inherently stable with reproducible behavior. Unless something breaks or changes, a given workload will run today as it did yesterday. To identify causes of changed workload performance, some vendors offer tools that capture change activity in the system.

If a sudden change in performance of a key workload is observed from day to day or (typically) between a Friday and the following Monday, it is very likely that something has changed in its environment. Possibilities include:

- Hardware—a CPU or channel path may be offline, or storage (central or expanded) may be different than it was at the previous IPL. While this might have been a far-fetched possibility in the past, a production partition in a logically partitioned system (IBM's PR/SM LPAR, HDS's MLPF, or Amdahl's MDF) can be changed all too easily during "test time" and not necessarily restored to normal before production resumes.

- Software configuration—MVS system parameters may have changed through IPL or operator action, JES initiators may be in a different configuration than they should be, or key data sets may have been moved or may have gone to secondary allocation with a gap between extents causing longer-than-usual SEEKs.

- Software—the possibilities are widespread.

- Application—the possibilities are even greater.

- Volume—transaction volume may have changed suddenly because of business changes not communicated to the data center staff.

Not all of these changes are detected by commercial change monitors. The available queries and reports should be examined, and the possibility of other changes should be considered as well.

9.2.6. A balanced methodology

A performance analyst can find value in any of the tools described above. A common pitfall to avoid is the Carpenter's Fallacy—if the only tool at hand is a hammer, every problem looks like a nail. For a performance management methodology to be complete, it must include a variety of tools to address each kind of problem typically encountered:

- **Undocumented or unexpected change:** Before wasting time going through a laborious process to determine the cause of a sudden change in performance, first use any means available to rule out a change in some factor that could affect the workload's computing environment.

- **Resource exhaustion:** Detecting resource exhaustion is an essential complement to capacity management. This condition is revealed by resource monitoring tools in conjunction with statistical analysis and graphical trending packages. Execution-state analysis tools can improve the accuracy of capacity projections by breaking down the response time (or elapsed time) trend for the workloads of concern into resource usage and delay components.

- **Mismanaged workloads:** A workload performing poorly is identified first by monitoring some measure of its service, such as response time. When a service exception is detected, execution-state data can then be used to determine the nature of the delay. Dominant states like "waiting for CPU" or "swapped by SRM—MPL" indicate that the workload might be mistuned in the system, and that a tuning opportunity exists. Such a determination is best done using smoothed historical data rather than real-time data. In the case of goal mode, this kind of mismanagement is typified by an inappro-

priately unfavorable goal or, in a constrained system, an importance level that is too low.

- **Chronic contention:** Continuing the methodology above, if the dominant state of the workload is one indicating contention, such as "queued I/O" or "waiting for ENQueue," tuning may be of little benefit. Instead, the workload should be monitored by a direct contention analyzer at previously characterized times of vulnerability. Once the contending workload is identified, it should be easy to decide how to eliminate or minimize the contention.

- **Acute contention:** Service analysis in real time can also be used to trigger the investigation of contention. It is less likely that system parameter errors will lead to a sudden erosion of performance than that some other workload is causing acute contention. For this reason, contention analysis is the technique of choice in the event of acute performance degradation, with execution-state analysis a secondary tool.

Practical examples of problem solving based on using these techniques and tools will be found in the last two chapters.

9.3. One Kind of Modeling

Modeling is a key tool of capacity planning. It is not widely used in diagnosing or resolving system or workload performance problems. Accordingly, a full discussion of modeling techniques and tools is not included in this chapter. However, we will consider an *ad hoc* approach to modeling DASD I/O, since it is very simple to implement with a calculator or a spreadsheet program, and can lead to useful insights about DASD performance factors. Another simplified modeling approach, popularized by Peter Denning and Jeffrey Buzen and used in several commercial modeling products, is Operational Analysis, an algebraic subset of queueing theory.[3]

An analytic model of a typical DASD I/O operation is relatively easy to construct, given the published facts about devices and their connections to systems, and measured data from performance monitors. Unfortunately, two factors work against apply-

3 B.Domanski, private communication.

ing this technique to the more sophisticated storage processor subsystems that are so widely used today:

- performance numbers just don't mean what they meant before. For instance, in a RAID 5 subsystem, a single I/O request may require data transfer to or from several different devices. For a READ, some of those transfers may be cached and others not cached.

- where data is readily measurable by the manufacturer, it may not be released. An example is protocol times, which tend to be the dominant component of I/O elapsed time in high-performance subsystems.

Consequently, the devices shown in the model results are not the most current. However, the techniques are described to a level which will allow the reader to apply the model to any device for which the numbers are published or measurable.

The first step in constructing such a model is to decompose the time span of the I/O operation into its component steps and to assert or derive a formula describing each component. Such a breakdown for a DASD READ or WRITE is shown in Figure 9-3.

When the formulæ include quasi-independent variables such as *path busy* or *device busy*, or assumptions such as the choice of a queueing model, the model should be evaluated for each meaningful value of such parameters. Predictions of overall service times or response times should then be checked or *calibrated*

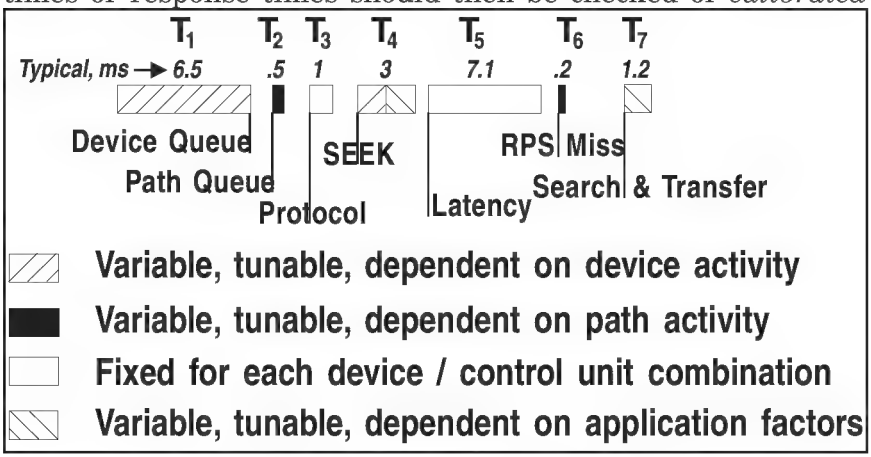


Figure 9-3. Components of I/O Response Time.

against actual experience and the model refined until it is valid for significant cases.

A significant pitfall in DASD modeling is to assume that the numbers reported for device and path utilization for a single system characterize the device or path, even though the device is shared across multiple physical, logical, or virtual systems. Especially when device queueing is to be calculated, only overall device utilization (from all sharing systems) is an accurate input.

Let us assume that we will find a block size of 4096 bytes useful for further analysis. We analyze a READ operation as follows.

9.3.1. T_1, UCBBSY delay

MVS will not attempt an I/O operation if the device is already known to be busy executing an I/O operation initiated by the current system. This condition is indicated by the "on" condition of a bit symbolically named UCBBSY in the device's *Unit Control Block* (*UCB*, a logical resource that MVS uses to represent the device). The operation is placed on a queue associated with the UCB until the previous I/O is completed.

UCBBSY delay cannot be represented by a simple formula. It is entirely dependent on the type, rate, and duration of the competing I/O activity on the device. Most authors assume that the competing I/O is independent of the operation being analyzed, and that the device has an exponential distribution of service times. These assumptions define an M/M/1 queueing model. Using that model, the expected value of the queueing delay is

$$T_1 = \frac{S_d \cdot B_d}{1 - B_d}$$

where

T_1 is the name we assign to this component.

S_d is the expected value of device service time.

B_d is the mean device busy fraction of time.

The M/M/1 assumptions do not hold up if competing activity is not independent, or if the service time distribution is more deterministic than an exponential distribution. Consider, for instance, a case in which a data set is being copied, block by block, to another data set on the same device. M/M/1 can be thrown out because its assumption of independence does not apply. The de-

lay can be calculated without resort to queueing theory and represents a worst-case example of *synchronous delay.*

As another exception to M/M/1, consider a dedicated paging volume responding only to single page faults. In this case, the M/M/1 model is too pessimistic, since the device service times are likely to be very consistent or *deterministic*, rather than exponentially distributed. In this case, another model, M/D/1, is more appropriate, and yields half the queueing delay of M/M/1.

Algorithm: In the cases presented later in this chapter, M/D/1 is used for the first two cases, representing near-idle and nominally loaded devices, while M/M/1 is used for the remaining cases of heavy loading and heavy loading with application-related perturbations.

9.3.2. T_2, initial path delay

We continue to assume that the CPU time component of I/O service time is negligible. The next opportunity for delay, after the device itself is found not to be busy from the current system, comes about when the selected path is busy as a result of another I/O operation. In MVS/ESA (or MVS/XA), the START SUBCHANNEL instruction always succeeds and the dynamic channel subsystem hardware tries and retries all defined and available paths until the I/O actually begins. When the path-busy condition can be attributed to busy control units, the queuing in XA takes place at the level of a *logical control unit* or *logical path*, a symbolic entity managed by the channel subsystem. This delay is known as *pending* (PEND) time in XA and ESA I/O reports.

Initial delay due to device controller (head of string) contention appears at this stage of the I/O operation, as does device delay due to use by another system (shared DASD contention). Estimating shared DASD delay requires knowledge of device activity due to all connected systems, and follows the same models as T_1 above.

However, since T_2 is usually insignificant, other than for shared DASD, it can be estimated according to a simple formula for single-path devices:

$$T_2 = \frac{B_p \cdot S_p}{2}$$

where

T_2 is the name we assign to this component.

B_p is the fraction of time that the path is busy, interpreted as a probability.

S_p is the expected value of a burst of path service time.

The probability-weighted service time burst is halved because the current demand for service is random with respect to the previous path activity. For a typical 3390 case, T_2 is less than half a millisecond.

With multiple paths, we have no simple estimator of the path-busy probability like the single path-busy fraction. If we regard B_p as a probability and the activity of multiple paths as independent, we can raise B_p in the formula above to a power representing the number of paths available for the I/O operation. It may be seen that T_2 is negligible for an I/O device using two or more paths.

Algorithm: The single-path formula is used for solid-state devices. For the DASD examples, all of which have four-path connections, the formula used is

$$T_2 = .5 \cdot S_p \cdot B_p^4$$

where S_p is the mean path service burst time and B_p is the mean underlying or *residual* path utilization.

9.3.3. T_3, initial CONNECT period (protocol)

Once the path is secured, CCWs specifying positioning actions are transferred to the control unit and then to the device controller for execution. During this time, path selection microcode in the storage director and device controller is executed, and dynamic allegiances are created between the channel and the storage director. The time used for all this intercommunication is known as *protocol* time and varies from device to device, from connection option to connection option, and even from microcode level to microcode level, but is typically one-half to 3 milliseconds. Protocol time for some control units is divided into two subcomponents, *overlapped* and *unoverlapped.* Unoverlapped protocol requires connect time that becomes visible as part of the device service time; overlapped protocol can be executed during device disconnect time but does contribute to the path-busy component.

We assume overlapped protocol of 0.5 millisecond and unoverlapped protocol of 1 millisecond for 9343s or 3990s controlling uncached 3380s or 3390s. For cached 3380s and 3990s the overlapped value rises to 1 ms. We use 0.5 ms each for unoverlapped and overlapped protocol of solid-state devices.

The manufacturers of solid-state devices typically state an "access time" of about 0.3 millisecond. This value is in addition to protocol. We also use an access time of 0.1 millisecond for a miss, and 0.5 millisecond for a hit in cache devices, the latter in addition to the base control unit protocol.

Algorithm: The foregoing protocol estimates are the author's alone, but they are based on published numbers for previous generations of controllers modified by the expectation of improvement in microprocessor cycle times and bandwidths.

9.3.4. T_4, disconnect for SEEK

When actuator motion is required, a SEEK CYLINDER device order is executed. During the time of motion, only the device is active, and the rest of the path is disconnected. (The allegiances established at path selection are retained, so that additional protocol time is not needed for the subsequent reconnect.) SEEK times are published for each device, but again, they must be interpreted according to the pattern of I/O activity in progress. For example, reading a half-track-blocked unfragmented sequential data set on a 3380 will require a repeated pattern of 30 no-movement SEEKs followed by a one-cylinder seek. If another data set is active on the device, however, this pattern will be broken up by SEEKs between the active data areas.

Device	Minimum	Average	Maximum	Typical
3380xJx	2 ms	12 ms	21 ms	3.6 ms
3380xK4	2	16	29	4.8
3390-1	1.5	9.5	18	2.85
3390-2	1.5	12.5	23	3.75
3390-3	1.5	15	33	4.5
9345-x1x	1.5	10	16	3.0
9345-x2x	1.5	11	20	3.3

Published "average SEEK" times represent seeking across one-third of the cylinders on the device. However, IBM has published measured results for typical application mixes, showing that a 30 percent incidence of "average SEEKs" is a good approximation to the mixture of no-movement SEEKs, minimum SEEKs, and average SEEKs experienced in those workloads. The table shows minimum, maximum, average, and typical SEEK times for a selection of IBM devices.

SEEK time is not present for fixed-head devices such as the IBM 2305, for solid-state devices such as the IBM RAMAC Electronic Array, and for READ hits on cache devices. For READ misses, the operation proceeds as for uncached DASD, with the details of the particular hardware determining whether data transfer to the channel is in parallel with cache staging or done as a subsequent operation.

Performance characteristics of WRITEs to cached devices are more highly dependent on hardware variations. If the cache is managed in the "store-in" manner, or if a separate integrity-write area of storage (IBM calls this *NVS* for *non-volatile storage*) is provided, WRITE hits are performed without a disconnect, followed by later asynchronous updating of the backing store device. With "store-through" cache management, the WRITE is not completed (the device remains busy) until the DASD is updated. In most cache designs, WRITE misses are equivalent to uncached WRITEs, but more responsive options are possible with nonvolatile buffering storage.

SEEK delay on DASD can be minimized with planning, and with subsequent tuning actions (when justified) because of delay to a key workload. For instance, several data sets required at the same time should not be on the same device if good response time is required. (We call this situation, when successive I/O to the same device are a consequence of the placement of data sets serving an application, *synchronous contention*.) In more complex data structures, multiple areas of a direct access data set or data base might be separated across volumes. Allocations of space on DASD should be compact; frequent routine maintenance using defragmentation utilities can hold down seeking due to the use of multiple extents as data sets grow.

When numerous small, active data sets with a high read-to-write ratio fill a high-capacity device, great benefit can come from attaching that device to a cached control unit. SEEK and latency delay is cut dramatically, and RPS miss is virtually

eliminated if it would have been a significant time component in the uncached configuration. As the price of memory declines, the use of cache has become far more general, approaching the universal on newer subsystems.

9.3.5. T_5, latency

Disk storage devices are not truly direct access or random access. There is an element of sequential delay while waiting for data on a track. The time spent waiting for the first byte of desired data (in fact the count field preceding the target record) to appear under the read head is called *latency.* Finding the correct record is done with a combination of SEEK and SET SECTOR, device orders established by separate CCWs. In most current device and control unit combinations, both of these commands are fetched together and communicated to the device as a combined device order. With newer control units that support the ECKD (Extended Count-Key-Data) set of commands, a combined operation called LOCATE RECORD may be used with the same effect.

With SEEK completed, latency begins. Plainly, the latency delay ranges from zero to the time taken for a full rotation of the disk. After the specified sector (circumferential or rotational position) is at the heads, the path must be reestablished so that the record may be located and the data transfer initiated. (The possible additional delays of the reconnect, also dependent on device rotation time as one factor, are covered under T_6 below.)

As may be seen in the modeled results, latency delay is usually the largest component of device service time. Considerable engineering investment has been made in an effort to minimize or eliminate latency delay. The following table shows typical latency numbers for devices that are commonly uncached. For cached devices and newer integrated subsystems including the various versions of RAMACs and EMC's Symmetrix, latency is reduced in two ways—through the near-universal use of cache and through use of more advanced devices with ever-smaller diameters and faster rotation times.

Device	Rotation Time	Nominal Latency
3380 (all)	16.56 ms	8.28 ms
3390-1,2,3	14.2	7.1

9345 (all)	11.2	5.6

An early engineering triumph over latency delay was achieved at high cost on the now-obsolete IBM 2305-2. Latency was reduced by use of a multiple requesting feature included with the device and its IBM 2835 control unit. The device appeared to MVS as a cluster of eight devices with successive addresses, and each such address or *exposure* could be driven independently. If requests are independent of each other (as for page-ins),[4] I/O delay may be reduced because the order of reconnect, and hence of I/O completion, is determined by the circumferential order of the desired data on the disk surface. An adaptation of the multiple exposure approach was used with the IBM 3880 Model 11 cached paging subsystem (another obsolete device), permitting multiple requests to be sent to the subsystem and letting hits be completed while misses were being serviced from the backing DASD.

Solid-state and cache devices eliminate or reduce latency much as they remove SEEK delay: No motion means no delay.

Algorithm: The simple half-rotation latency algorithm is used for uncached DASD. It is multiplied by the complement of the hit ratio for cached DASD, and is zero for solid-state.

9.3.6. T_6, RPS miss (reconnect) delay

Disconnected Command Chaining (DCC) was introduced along with block multiplexor channels and the IBM 2314 DASD in the late '60s. Its purpose was to increase the beneficial utilization of scarce and expensive channels and control units by allowing the active path elements to be freed while slow, device-specific operations could proceed on their own. SEEK was an obvious first candidate.

The latency period, half a rotation on average, was a connected operation; SEARCH requires the entire path from channel to device. When the SEARCH was satisfied, data transfer proceeded with no further delay. Why, then, was an extension of DCC needed? Remember that channels were expensive at that time

4 A page-in to resolve a page fault has no related successors, since the address space incurring the page fault cannot proceed and thus could not have incurred a second fault. However, since SP 4.2, page faults are resolved through block page-ins, so some succeeding faults may be avoided. Even so, the next page fault that requires a page-in from auxiliary storage is again unrelated to its predecessor. There are just far fewer of them.

and that SEARCH required the channel. Suppose that the disconnect continued after the SEEK was completed, extending into a device-specific simple operation to detect a particular angular position. Reconnect could then take place, followed by a much-reduced SEARCH (still required to establish orientation for the following READ) and then the data transfer operation. The new function Rotational Position Sensing (RPS) was introduced, but along with it came an unwanted new phenomenon.

The saga of RPS and RPS miss thus began when DCC was extended to include not only SEEK, but latency as well. A new device order, SET SECTOR, was added to the channel programs of standard access methods. The argument of the SET SECTOR order was calculated so as to allow reconnect to take place a few milliseconds before the data passed under the read/write head.

DCC for SEEK was unconditionally helpful. When the SEEK was completed and the device was ready for reconnect, no further delay was possible; the device waited patiently for the path, and the subsequent SEARCH and data transfer were then executed. In the case of the RPS reconnect, however, the timing was critical; a missed reconnect (for instance if the path was busy when rconnect was attempted) meant that the next try could come only after a full rotation of the device.

RPS did succeed in improving the effective data transfer rates of channels and storage directors. Beneficial utilization went up; overall usage went down. The price for the improvement, however, was the creation of a new kind of performance degradation.

If we make the queueing-theory leap of faith, that the average path utilization (U_p) over some interval is a good approximation to the probability that the path will be busy at some time of interest, we see that the probability of an RPS miss is equal to that utilization, and that the magnitude of the miss is exactly one rotation period. Subsequent misses are possible, with probabilities given by successive powers of U_p. This seems simple enough.

But it isn't. The path utilization of interest is not the entire utilization, but rather what Tom Beretvas calls *residual path busy* (RPB)—path busy with the activity of the device being analyzed discounted or removed. In the formulæ for RPS miss, B_p is in all cases RPB, given by the formula

$$B_p = \frac{U_p - U_d}{1 - U_d}$$

where U_p is the path utilization and U_d is that part of the path utilization due to the current device. It may be seen that RPB is zero for a device on a dedicated path, reminding us that devices on dedicated paths cannot experience RPS miss.

The formula for the simple case of RPS miss on a single path is thus

$$M_e = R_d \cdot (B_p + B_p^2 + \ldots)$$

where M_e is the expected value of RPS miss, R_d is the device rotation period, and B_p is residual path busy. In closed form, the formula is precisely the M/M/1 queueing formula for delay in a queue to a single server with exponentially distributed service time R_d and utilization B_p:

$$M_e = \frac{R_d \cdot B_p}{1 - B_p}$$

In MVS/370, path elements could appear to be busier with respect to a given device than indicated by overall utilizations; multiple paths in MVS/370 improved availability at a cost in performance. An *effective path busy* correction (increasing the apparent path utilization to account for the performance loss caused by multiple paths) was needed in addition to the residual path busy correction. The net effect in MVS/370 was that RPS miss (a function of effective path utilization) became a source of significant I/O performance degradation, and a prime problem to be solved in the process of I/O performance analysis and tuning.

With Extended Architecture, the beginning of a solution was at hand. First, the 3380, with dynamic pathing, was announced, before the XA announcement. The 3380-AA4 was initially announced as a device that offered only Dynamic Path Selection. This feature seemed at first (before XA) indistinguishable in practice from String Switching, available on the IBM 3350 and a source of performance degradation as the price of an improvement in availability.

When XA was announced, the full dynamic pathing capability was revealed, and the shrewd analysts who recommended 3380-A04s (without dynamic pathing) were considerably embarrassed. Dynamic Path Reconnect (the other part of dynamic pathing) exploited two paths at first, cutting RPS miss drastically. (The probability of the first miss was now roughly comparable to that of the second miss in MVS/370.) With the 3990 control units and attached 3390s or 3380 J and K models, four paths are supported, nearly eliminating RPS miss at moderate

path utilizations. Four-path operation also is supported on the downsized and lower-priced IBM 9345 DASD with the 9343-C04 or -D04 Storage Controller, and on all of the more recent devices including the entire RAMAC family.

Contemporaneous developments also reduced the magnitude of the RPS miss problem. Devices that do not disconnect from the path have no reconnect problems; solid-state devices, and cache devices encountering hits, do not disconnect. Channels are now relatively cheap, fast, and abundant, so there is less need to multiplex on individual channels. With the 3990 or the integrated control unit of the 9340 or the RAMAC subsystem, individual control units are less expensive as well. Four-path logical control units (LCUs) reduce RPS miss to insignificance, and a new generation of DASD and control units can now use local storage to complete the job.

As we view the rise and fall of RPS miss, we might wonder: Is RPS needed any more? Are average channel utilizations now low enough and four-path connections now common enough to render connected SEARCHes a nonproblem? Can further improvements in outboard intelligence render connected SEARCHes unnecessary? Can a device (and possibly its head of string) handle a new kind of SEARCH and data transfer without rotational delay beyond nominal latency? Can read-head buffering eliminate some part of latency as well? On the other hand, does the growing use of cache control units and four-path connections provide another way to render RPS miss a curiosity of the past?

If the next generation is one that avoids RPS miss, its initial device was Storage Technology Corporation's 8380RQ, a 3380-class device with *actuator-level buffering* (ALB™). The device featured a single track-size buffer which started to fill as soon as SEEK was complete. Latency was the only rotational delay. SET SECTOR took on the character of SEEK: the device waited for the first time the path was available, then data transfer proceeded without need for synchronization. Because the buffer was limited to a single track, multitrack operations (as for block paging) could suffer a second latency delay, or the microcode might have suppressed buffering for multitrack operations. A second track buffer could have made sequential exchange buffering possible.

Track buffering was good, but buffer storage in the control unit is better. The ALB device was still limited to synchronous data

transfer after the first track. With a local buffer in the control unit, the IBM 3990 and 9343 controllers finally allowed the data to wait for the path. The buffer storage is not cache. It is managed on a simple LRU basis and yields several benefits:

- it eliminates RPS miss by allowing immediate data transfer from device to the buffer regardless of whether the channel is busy
- it allows data transfer from the device to the control unit to proceed independently of data transfer from control unit to channel
- it allows data transfer to the channel at the full data rate supported by the channel and control unit regardless of the device speed

These characteristics, as well as some command sequence and protocol considerations, define *nonsynchronous* DASD. The nonsynchronous attribute has become a standard part of devices since its first appearance.

Algorithm: The definitive model for a process such as RPS miss in a four-path environment is the computationally complex M/M/4 queueing model. A slightly more pessimistic but much simpler representation is available with the assumption that mean path utilization (always adjusted as residual path busy) is a probability of finding the path busy, and that the probabilities associated with each of the four paths are independent. With this assumption, the simple formula is

$$T_6 = (1-H) \cdot R_d \cdot B_p^4$$

where **H** is the cache hit ratio (zero for uncached cases), R_d is the device rotation time, and B_p is the mean residual path busy for the four paths.

However, for the devices and control units that might now be considered current the discussion is irrelevant. RPS miss is gone, and we can drop it from the model. T_6 is so close to zero that we ignore it in each case we consider.

9.3.7. T_7, SEARCH and data transfer

Once reconnect is achieved, the remainder of the I/O operation is dependent only on the physical characteristics of the

device. For solid-state devices and for cache READ hits, a minimal access delay is followed by data transfer at the rated device-to-channel speed. Thus our 4096-byte transfer will take 1.37 milliseconds at 3 megabytes per second or 0.91 millisecond at 4.5 megabytes per second. For ESCON channels transferring data at the original 10 MB/s rate or the enhanced 17 MB/s rate, the data transfer is 0.41 ms or 0.24 ms, respectively. Access delay is already included in the protocol delay (T_3).

Algorithm: On "real" DASD, SEARCH time must be added. If we note that ten 4096-byte records can be contained on a 3380 track, it seems right simply to assume that finding and transferring one of them takes no more than one-tenth of a device rotation period, or 1.66 milliseconds. The SEARCH time is absorbed in this approximation. Similarly, for 3390s, which can contain 12 4K records per track, the 14.2-ms rotation time is divided by 12, giving a page transfer time of 1.18 milliseconds. For 9345s, the computation is 11.2 ms (rotation) divided by 10 records per track, giving 1.12 ms per page.

For solid-state devices and cache hits, the computation is more direct, as noted above.

9.3.8. Analysis of modeling results

We'll now examine the results of putting together the components of DASD I/O response time under various conditions of activity, overhead, and interference.

To define a case for modeling, it is necessary to assume a target level for both device and path utilization. Usually one of these elements (most frequently the path) will require assuming an underlying utilization level in order to reach the target. If the model is being used to analyze a real system, measurement data can supply precise starting values.

The modeled quantities T_2 through T_7, and finally T_1, are then calculated and accumulated to arrive at estimated response times. Throughput values and effective data transfer rates can then be computed.

Several cases of single-page (4096 bytes) READs on various devices in MVS/ESA environments are examined below. XA environments are equivalent to ESA in this context.

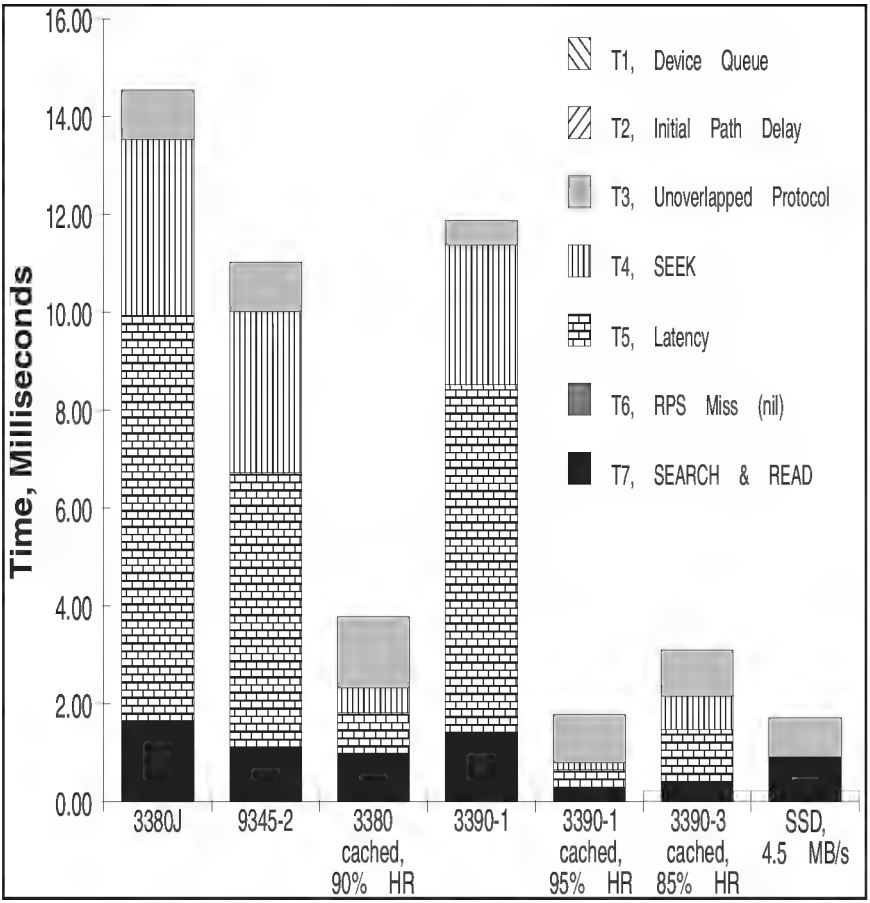


Figure 9-4. Modeled Results, No Load.

The first case represents best-case performance. No competing device activity is assumed, and the entire path to the device is assumed to be only 10 percent busy from other devices. Although not necessary to this case, all devices except solid-state are assumed to have four-path connections. (No I/O rate appears in Case 1; these are response times for isolated I/Os on idle devices.)

Let's examine these results more closely to understand the contribution of each component to response time. Figure 9-4 shows those components for a selection of devices and modes.

Note that SEEK and latency predominate for uncached DASD, while protocol and the combination of SEARCH and data trans-

fer are the main components for cached devices, and are the only components for solid-state devices.

The conditions in the "Notes" column apply as well to the corresponding lines in each of the other cases.

Case 1—Unconstrained Device Performance			
Device	**Director**	**Response Time**	**Notes**
3380J	3990-2	14.9 ms	4-path base case
3380K	3990-2	16.5	Longer SEEK time
3380K	3990-3	3.9	90% hit ratio
9345-B12	9343-C04	11.0	
9345-B22	9343-D04	11.3	
3390-1	3990-2	12.2	
3390-1	3990-3	1.8	95% hit ratio, 17 MB/s ESCON
3390-2	3990-2	13.1	
3390-2	3990-3	2.5	90% hit ratio
3390-3	3990-2	13.9	
3390-3	3990-3	3.2	85% hit ratio
SSD	n/a	2.2	3 MB/s
SSD	n/a	1.8	4.5 MB/s

The second case represents "normal" device and path loading according to commonly accepted rules of thumb. Because we are assuming uniform block size and minimal seeking, it is appropriate to use the M/D/1 queueing formula to estimate device queueing delay. Device utilization is held to the rule-of-thumb limit of 30 percent. Total path utilization is held to the common System/370 guideline of 30 percent. Path utilization for cache and solid-state devices is 50 percent. All uncached DASD cases assume four-path operation.

We won't take a detailed look at Case 2, since the same effects are present in exaggerated form in Case 3.

Case 2—Moderately Loaded Devices				
Device	**Director**	**I/O Rate**	**RT**	**Data Rate**
3380J	3990-2	21.4 /sec	17.1 ms	88 KB/sec
3380K	3990-3	90.4	4.0	370
9345-B12	9343-C04	29.3	12.4	120
9345-B22	9343-D04	28.5	12.8	117
3390-1	3990-2	26.0	14.0	106
3390-1	3990-3	166.6	2.2	683
3390-2	3990-2	24.1	15.1	99
3390-2	3990-3	123.5	3.0	506
3390-3	3990-2	22.7	16.0	93
3390-3	3990-3	96.0	3.8	393
SSD (3.0)	n/a	129.1	2.8	529
SSD (4.5)	n/a	163.6	2.2	670

The next case represents heavier loading of devices and paths. We assume the devices to be 50 percent utilized, but with exponentially distributed arrival rates and service times. M/M/1 queueing is thus assumed. Total path utilization is assumed to be 40 percent for DASD and 80 percent for cache and solid-state devices.

Case 3—Heavily Loaded Devices				
Device	**Director**	**I/O Rate**	**RT**	**Data Rate**
3380J	3990-2	35.6	28.1	146
3380K	3990-3	140.8	7.1	577
9345-B12	9343-C04	48.9	20.5	200
9345-B22	9343-D04	47.5	21.1	194
3390-1	3990-2	42.9	23.3	176
3390-1	3990-3	255.0	3.9	1044
3390-2	3990-2	39.9	25.1	163
3390-2	3990-3	193.3	5.2	792
3390-3	3990-2	37.6	26.6	154
3390-3	3990-3	138.7	7.2	568
SSD (3.0)	n/a	201.4	5.0	825

Case 3—Heavily Loaded Devices				
Device	**Director**	**I/O Rate**	**RT**	**Data Rate**
SSD (4.5)	n/a	256.1	3.9	1049

Let's see now how the details have changed compared with those of Case 1. Figure 9-5 shows the component breakdown of device response time under heavy load.

Device queueing (T_1) has become the largest component, equal to

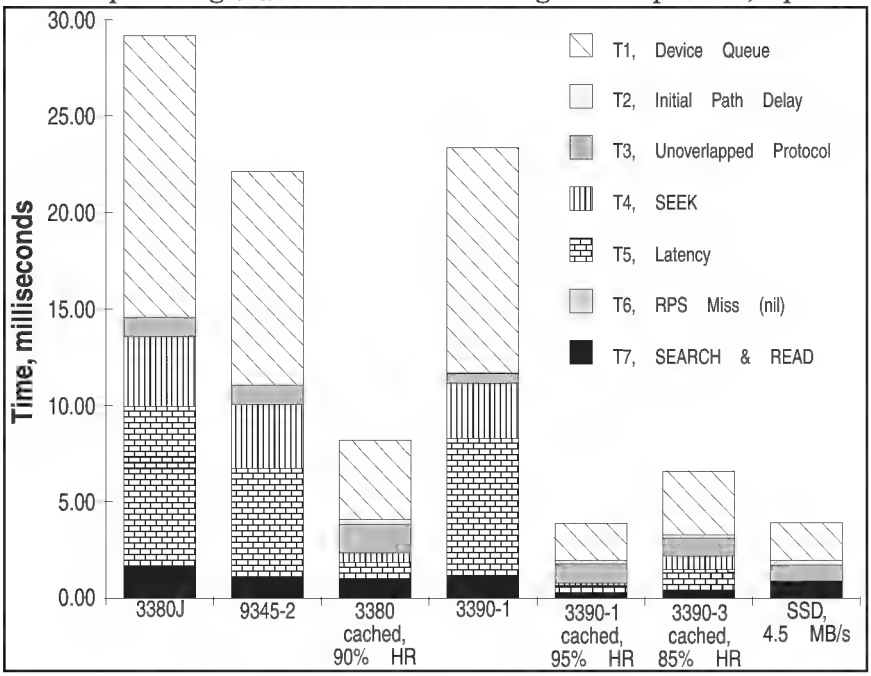


Figure 9-5. Modeled Results, Heavy Load.

all the other constituents of response time. Even under these load conditions, cached DASD (even with a relatively low 85 percent READ hit ratio) and solid-state devices deliver response times of less than 10 milliseconds.

In the three preceding examples, we see the expected progression of degradation with increasing I/O rate and throughput. However, it's not a great illumination. From unloaded to heavily loaded, response time more or less doubles, mostly due to queueing. As we drive the devices into heavier loads yet, we can expect to see yet another doubling of response time when utilization approaches 75 percent.

But everyone has seen much worse performance than that. Are the modeled results wrong?

No, but the cases are incomplete. We need to add two more cases, reflecting the susceptibility of uncached DASD to "arm stealing" and the susceptibility of all the modeled devices to contention from workloads on other systems. The first additional case assumes that 80 percent of I/Os are corrupted by the addition of an extra "average" SEEK, typical of experience where a single data set is split across discontiguous extents, or when an application incurs "synchronous delay" because of placement of multiple data sets on a volume. Note that solid-state devices are completely unaffected by this scenario and that cached DASD with high hit ratios are only slightly affected.

Case 4—Heavily Loaded plus Synchronous Delay				
Device	**Director**	**I/O Rate**	**RT**	**Data Rate**
3380J	3990-2	23.5 /s	42.5 ms	96 KB/s
3380K	3990-2	19.6	51.2	80
3380K	3990-3	98.9	10.1	405
9345-B12	9343-C04	30.6	32.7	125
9345-B22	9343-D04	29.1	34.4	119
3390-1	3990-2	28.6	34.9	117
3390-1	3990-3	179.2	5.6	734
3390-2	3990-2	24.9	40.1	102
3390-2	3990-3	129.4	7.7	530
3390-3	3990-2	22.5	44.4	92
3390-3	3990-3	97.5	10.3	399
SSD	n/a	201.4	5.0	825
SSD	n/a	256.1	3.9	1049

In the final case, we add to our scenario the type of perturbation experienced frequently when DASD is shared between systems. Shared DASD contention once was dominated by device RESERVEs: The device was tied up by the other system for a relatively long duration. The increasing use of GRS and functionally similar packages has diminished the number and duration of RESERVEs

dramatically, but active I/O from a busier source of I/O requests can cause numerous periods of elongated PEND time; the system without an active channel program must wait for the one that has the device. If the I/O from the other system is of long duration (such as a search of a partitioned data set directory), the delay can be just as disruptive as RESERVE delay used to be.

In this case, we assume that one I/O per thousand must wait for 10 additional seconds as a result of shared DASD bursts of activity. If this duration of disruption seems excessive, just think of it as a 1-second delay 10 times as often. The averages will be the same, but the variability will be reduced by a factor of 10.

Case 5—Case 4 plus Shared DASD Contention				
Device	**Director**	**I/O Rate**	**RT**	**Data Rate**
3380J	3990-2	16.0 /s	62.6 ms	65 KB/s
3380K	3990-2	14.1	71.2	58
3380K	3990-3	31.3	32.0	128
9345-B12	9343-C04	19.0	52.7	78
9345-B22	9343-D04	18.4	54.4	75
3390-1	3990-2	18.2	54.9	75
3390-1	3990-3	39.6	25.3	162
3390-2	3990-2	16.6	60.1	68
3390-2	3990-3	36.5	27.4	150
3390-3	3990-2	15.5	64.4	64
3390-3	3990-3	33.5	29.9	137
SSD	n/a	40.6	24.7	166
SSD	n/a	42.2	23.7	173

Note the uniformly hideous response times and diminished throughput, even for solid-state or cached devices which are otherwise resistant to increasingly stringent I/O activity. The statistics in the tables do not reflect the true nature of the shared DASD contention problem. It is not the degraded average response time that is so troublesome; it is its variability that translates to poor application service.

9.3.9. Conclusions from modeling

A few observations can be made from these modeled results:

- Cache and solid-state devices suffer far less response-time degradation with increasing load than do pure DASD, even with much higher I/O loads.
- Buffered data transfer, characteristic of nonsynchronous DASD, helps to stabilize response times under heavy load, even without caching.

9.4. Summary of Measurement and Modeling

A key need in understanding and implementing MVS performance management is to be familiar with readily available tools for measuring all aspects of system and workload performance. Whichever tools are at hand are often used only occasionally, when acute problems occur. In the confrontational environment often encountered by performance analysts, learning a tool "under the gun" is not a good career choice.

If service level targets are agreed upon by data center and user management, measurement to those targets is the first necessary step. (The importance of service targets cannot be overemphasized. If you're aiming at nothing, that's what you'll hit.) While the targets are being met, using an historical execution-state analysis tool can help the analyst understand impending constraints and assist in refining capacity projections to the extent of determining when CPU or storage upgrades are needed.

When targets cease to be met consistently, the type of incident determines the time span of the tool to be used. A gradual and consistent erosion is usually best analyzed with an historical tool. Sudden loss of a target, on the other hand, should be diagnosed with a real-time monitor. (Such a sudden change in performance should be examined with a change monitor as well. A system or application does not wake up and call in sick. If performance changes, something has changed in some other factor: hardware, system software, application software, volume, system parameters, etc.)

Modeling is usually used to aid in capacity planning. An *ad hoc* form of analytic modeling can be used to make performance predictions or choices, or to analyze DASD performance when DASD represents a significant bottleneck.

9.5. Chapter Questions

1. Review the use of measurement and monitoring tools in your installation. Do you have appropriate tools to deal with each class of problem examined in this chapter?
2. Describe the steps you would follow to determine the cause of a sudden loss of performance in the most important workload in your system. How would the availability of other tools aid in your diagnosis?
3. Do you make regular use of the tools you have? If not, are you planning to learn how to use the tools in a problem-solving context?
4. Examine RMF device activity and related reports for representative DASD in your system. Can you find the data (in the device, LCU, and channel reports) to follow the composite modeling methodology in this chapter? Plug in the numbers and examine the results. Do the modeled service times and response times match the measured ones?
5. For devices not tracking to the model, determine what factors might contribute to the anomalies. (*Hint*: Look at shared DASD contention.)

Chapter

10

Solving Performance Problems

So far, we have examined MVS's resources and the factors that affect the performance of workloads. In the previous chapter we looked at ways to monitor resources and workloads and to model the behavior of conventional DASD I/O. We thus have the means to identify service problems. In this chapter we will consider ways to anticipate and prevent performance problems, and define further the steps that may be taken to solve problems found by monitoring.

10.1. Lost Opportunities

At many installations, MVS is received and installed in preconfigured offerings. These have been known by such names as IPO, MVS Express, CBDPO, and CBIPO. The newer service or system replacement offerings from OS/390 (preconfigured CD-ROM for R/390 and P/390 systems, ServerPac, and CBDPO) continue in the same vein. While these offerings simplify greatly the process of installing MVS and keeping it current, they necessarily bring with them a highly constrained hardware environment. Getting MVS "up and running" is reduced to restoring two or three DASD volumes and then IPLing from one of them. Those volumes include all of the MVS parameter, procedure, and load libraries; JES SPOOL space and the JES2 CHKPT data set; page data sets; SYS1.LOGREC; and SMF data sets.

In a new MVS system, proving the viability and basic functionality of the system and merging applications with it are activities that correctly receive higher priority than adjusting the system for good performance. It is easy to take the position that adjustments can wait until the workload has built up and the first problems appear.

IBM is not responsible for early inattention to performance management. Spelled out in the online and printed documentation for IPO and its successors including OS/390 are explicit and frequent admonitions to distribute the key system and subsystem data sets across multiple paths and volumes. The harried system programmer, however, remains too busy with functional issues to be concerned with reading such publications, let alone responding to their prescriptions. In practice, therefore, performance problems tend to appear as the production workload is applied.

10.1.1. A typical scenario

Consider the appearance and evolution of "paging problems" in a typical mixed-workload new MVS/ESA system that has no expanded storage, running in Workload Manager compatibility mode. There is a gradual build-up of TSO users and the batch jobs that they submit. At the same time, more and more users log on to CICS and soon produce a significant transaction rate. What happens all too frequently is that at some time during the build-up, CICS response time starts suffering and TSO users begin to experience intermittent long delays.

What's happening is that MVS's default real storage management strategy is being allowed to run without installation-specific control. That strategy has these elements:

- As the multiprogramming level (MPL) increases, the working set of active address spaces (initially just the aggregate of allocated frames) increases, depleting the available frame queue (AFQ). When the AFQ goes below 150 frames, page stealing commences.
- MPL will increase until any one of these defaults is crossed: The average system-wide high UIC becomes less than 4, the CPU is fully utilized and at least one address space has not been dispatched during an SRM RM2 interval, 82 percent of central storage with real address below

16 megabytes is fixed, or 66 percent of all central storage is fixed.

- Another influence leading to depletion of the AFQ is MVS's moderate default bias in favor of logical swapping. TSO users idle for up to five seconds will be logically swapped. Even though the AFQ may be empty, logically swapped address spaces will not become eligible for physical swaps until a logical swap duration exceeds the system think time, plus a release-dependent "grace period."[1]

- Similarly, MPL will not be reduced until the system-wide high UIC is less than 2, or 88 percent of central storage with real address below 16 megabytes is fixed, or 72 percent of all central storage is fixed.

- Address spaces (other than TSO users in a rapid succession of trivial transactions) will be subject to delay from page fault processing, caused by subsequent reference to pages previously stolen.

What's wrong here? Nothing! MVS is acting correctly in accordance with the low-level, resource-oriented view of compatibility mode, and is doing exactly what its default parameters tell it to do. There are two complementary mechanisms to control the available frame queue. The micro-level control, page stealing, operates without the need for direction and is always active. Macro-level control is done by MPL adjustment. However, the MPL adjustment mechanism is to all intents and purposes crippled by default. More precisely, it is not responsive to any control variable related directly to real storage constraint. Compounding the potential for problems, the default control of logical swap eligibility is strongly influenced only by UIC (if less than 30) and only weakly controlled by the amount of storage already allocated.

As pointed out in Appendix B, RCCPTRT is the OPT parameter most appropriate for MPL adjustment, but it is shipped disabled. Simply beginning to use this parameter can overcome most initial "paging problems." In this chapter we'll describe

1 System think time by default (in SP 4.2) ranges from zero to five seconds, starting at the high value and not moving down until the UIC goes below 10. The grace period was 15 seconds for MVS/370 and XA releases prior to SP 2.1.7. Since then it has been 2 seconds.

setup activities that should prevent these kinds of avoidable performance problems.

Note that with expanded storage, the system is much more capable of defending itself from paging delay. However, there are still default pitfalls: pages of address spaces with different importance to the installation go to expanded storage without priority differences, and expanded storage can be depleted leading to unpredictable and disruptive periods of migration. Paging delay then hits the system "out of nowhere," affecting all workloads as pages move from expanded into central and then back out to auxiliary storage. The effect of central storage depletion is an increased page movement rate to and from expanded storage. The symptom of such an increase, even without increased paging to and from auxiliary storage, is an increase in unproductive CPU utilization. This increase in turn triggers the Working Set Manager of the SRM to take action against specific address spaces to hold down the paging rate or make more effective use of central storage.

The foregoing horror story recitation should help to strengthen the commitment of anybody with doubts about the worth of going to goal mode. All of these considerations and pitfalls about managing paging and MPL vanish in goal mode; all goal mode needs is an approximately correct service policy and adequate storage and paging configurations.

10.1.2. Consequences of neglect

While it is possible to recover after the fact from most performance problems, being constantly in "react mode" takes its toll. For most performance anomalies, the threshold of notice is much lower for the direct users of the system than it is for those managing it. As response time of a key workload becomes occasionally unacceptable and then consistently substandard, there will already have been a flow of complaints from the most aware (and probably the best informed) of the users. If the data center staff is doing nothing to monitor the quality and consistency of delivered service, the complaints come as a surprise, and the credibility of both those reporting the problem and those accused of causing it comes into question.

As the complaints mount, they will tend to be escalated to higher levels of management. In many cases, the ego-involvement and *ad hominem* character of the dialogue worsens until

issues reach executive levels. Only then might a decision be made to conduct an "objective" investigation and create a plan to solve the current problem. The task force (the formation of which is a very likely development) may call in a consultant or rely on the hardware vendor's marketing support staff for advice. There is little doubt that money will be spent, and that the data center staff will have a solution imposed on them that further attacks their credibility.

It doesn't have to be this way. The critical failing that led to this sorry sequence of events was failure to monitor delivered service for quality and consistency. In many installations, more is spent on resource usage accounting systems that capture usage data and apportion costs to the users than on performance monitors and staff to ensure consistent user satisfaction.

In an installation where customer[2] satisfaction is a genuine goal, the cost of tools to measure, track, and report on service and to help solve problems rapidly is paid willingly. In turn, the cost of such tools is more than paid back by eliminating the *ad hoc* emergency measures put together for dealing with problems that are ignored until they have escalated.

At the risk of belaboring the obvious, the ugly scenario is much less likely to occur if Workload Manager goal mode is operational on all the MVS systems with significant business-critical workloads.

10.1.3. A better way

With a proactive attitude regarding performance management, a data center staff can expect a cooperative relationship with its customers, rather than the all-too-frequent adversarial one. When the data center and those it serves agree on the volume of demand, on performance targets, on the methods of measurement and reporting, and on the quality of service shown by those measurements, the benefits can be considerable. The data center staff gains a reserve of credibility and can use that credibility to obtain well-thought-out business volume planning information. The resulting data center capacity plans and their embodiment in hardware come far closer to what is optimal for the

2 Substituting the word "customers" for "users" is an important consciousness-raising device for the data center staff in pursuit of quality service.

business than plans developed in a grudging atmosphere of mutual distrust.

10.1.4. Picking up the pieces

Although it is more difficult to repair a reputation than to build one in the first place, such a step is essential when organizing the performance management function in an installation that has not had one in the past. Where there is not sufficient mutual trust to elicit agreement on service targets, the repair process may take even longer. A strategy that has worked is for the data center staff to begin publishing (to management in general) frank reports of the data center's own targets and attainment of those targets. After a few months of such unilateral performance reporting, the results take on the character of *de facto* service targets. When the customers begin holding the data center to its own published targets rather than to some vague and subjective measure of acceptable service, the barriers begin to come down. In a goal mode environment, this process is akin to operating at first with something like Cheryl Watson's *Quickstart Service Policy*, where the goals are set arbitrarily by the data center staff—at first.

The fragile measure of trust built after some time in this remedial mode must be reinforced by a consistent orientation to providing acceptable service and accepting, and being responsive to, every objectively provable complaint from a customer. By now it should be obvious that monitoring and measuring with the appropriate tools is an indispensable part of such a program.

10.1.5. How problems arise

Computing systems, especially MVS systems, are generally stable and determinate. Hardware components rarely fail, and the system's hardware behavior is unlikely to change without apparent cause. Somewhat more often, an MVS system software component goes wrong, based on an exceptional input or a rare combination of values. Operating system defects usually lead to ABENDs rather than to performance anomalies. Similarly, subsystem and application program defects usually lead to functional failures.

How, then, do performance problems arise? If not random factors or out-of-the-way program bugs, what will cause a production job to complete within 25 minutes on Tuesday just as it did

the thirty previous runs, and then take 2 hours to complete on Wednesday? Why should TSO trivial response time, previously stable at half a second, increase to 2 seconds some Monday morning?

The answer is usually unremarkable: Something has changed. Perhaps it was the workload. However, the operations production control staff will know if there is an exceptional increase in volume for a key application, and an unplanned sudden increase in the number of logged-on TSO users is hardly likely.

What Changed?

If we can rule out workload change or random influence as the likely cause of a performance problem, we must conclude that something changed in hardware, software, operating system parameters, or the operating environment of the affected workload.

Most installations have more or less structured change management systems. Such systems tend to concentrate on keeping track of change plans based on what people say, and do not include a means of verifying that an asserted change really occurred.

Access control systems such as IBM's RACF and Computer Associates' Top Secret and ACF2, with logging options active, can keep track of access for update to key data sets. However, they do not intervene in hardware changes, nor can most systems bear the overhead of their logging accesses to all potentially troublesome data sets.

A change monitor may provide additional valuable information. These programs record and detect status and change information about various hardware, operating system, and data set elements, enabling them to subsequently report on what changed within a time frame.

The combination of change management systems, access control systems, and change monitors can provide the information needed to identify the factors that cause performance changes.

10.2. Configuring for Performance

The proactive approach to performance management begins with configuring the hardware and its workloads. Stability depends on being sure that unplanned changes or unforeseen consequences of planned changes can be identified rapidly using

the approaches suggested above. The following topics spell out other actions to be taken in advance of problems appearing.

10.2.1. Complementary workloads

One of the common mistakes made in setting up MVS systems is to assume that different kinds of workloads together on one system will inevitably interfere with each other. In the interest of preventing such interference, some choose to set up one-dimensional systems with, for instance, one or more systems running only a CICS workload and all TSO on another system.

If the CICS-only system is also production-only, it's difficult to derive full benefit from the CPU resource. CICS internal queueing will cause considerable delay to transactions ranking low on CICS's internal dispatching queue, at the level of about 70 percent uniprocessor utilization. If the system is a multiprocessor, a highly subdivided CICS MRO system is needed in order to take full advantage of all processors.

Suppose, however, that we had loaded the system with CICS to a level of 60 percent CPU utilization (representing CICS and MVS overhead, monitors, and necessary supporting subsystems), then added batch to fully load the CPU. If the batch is well behaved (that is, if it does not make unusual demands for real storage, perform inefficient I/O, or interfere with CICS's I/O devices or ENQueue names), CICS performance should be close to what it could be in a one-dimensional system loaded to about 70 percent CPU utilization.

If there is a clear hierarchy of importance and time urgency in the overall array of MVS work in a data center, it should be possible to load each system to a level just short of that at which internal queueing causes perceptible delay in the highest-urgency workload, then to bring the system up to nearly 100 percent utilization with low-urgency work. When doing this, it is important to recognize contention possibilities, taking steps to avoid inherent contention in advance and monitoring key workloads' performance to detect operational contention.

A contention pitfall, related to the idea of setting up one-dimensional systems, is the belief that a development subsystem should be isolated from its corresponding production subsystem to the extent of placing each on its own system or LPAR. While

there can be justification for such separation based on integrity considerations, there is often a severe performance penalty.

If the entire production environment were duplicated on the secondary system, there would be little potential for interference. In practice, however, some libraries are left on the production system, with shared access from the development system. Since the purpose of the development system is to bring about change in the production environment, it is almost inevitable that library updates can tie up production libraries. Even if the production libraries themselves are not updated, updates on the same volume can cause performance degradation due to excessive SEEK activity as well as periods of RESERVE delay during updates of volume-specific information. If, on the other hand, production and development run on the same system, member-level updates are much less disruptive, and the impact of development I/O activity is much easier to measure and deal with. Once again, if workloads are split across systems, make sure the split is as complete as possible.

Relief for such contention problems is much more easily achieved if the systems are configured in a parallel sysplex. As the remaining access methods become enabled for data sharing, fewer instances of one system delaying another will be likely.

10.2.2. How much real storage?

With Gary King's *Workload Characterization* and *Processor Storage Estimation* methodology available in various forms from IBM, SHARE, GUIDE, and CMG,[3] there is enough guidance on establishing a correct real storage configuration. King's methodology is clear and simple, and should be used routinely to check and track workloads against configurations.

The essence of King's approach is a combination of measurement, engineering analysis, and economic analysis. RMF measurements supply workload-specific values for various categories of storage use. For measurement accuracy and repeatability, central storage in each system must be placed in a slight state of constraint to ensure that page stealing works to keep measured frame counts close to true working sets. The measured results supply the intermediate points on a series of graphs that depict

3 King's papers may be found in the 1990–1992 *Proceedings* of SHARE, GUIDE, and CMG. His IBM *Technical Bulletins* may be obtained through local IBM offices.

a miss-ratio or hit-ratio curve for each storage use category. Extreme values are based on "all or none" assumptions for storage availability.

King then asks the reader to choose the degree of I/O avoidance desired for each category, suggesting that goodness lies on the flat (near-horizontal) part of the curve in each case. Based on such a choice, he goes on to build up a table of required processor storage for categories including the system, subsystem address spaces, active address spaces, idle address spaces, load libraries, and data sets. Summing across the table and adding a safety factor leads to a total of required processor storage to accommodate the current or planned workload. The final choice in King's process is to select the balance between central and expanded storage based on economic considerations, trading a cost increment against the CPU time needed to move pages.[4]

One way of looking at central storage without expanded storage is simple: There should be enough real storage to allow the CPU to become fully utilized with acceptably low paging delay in all key workloads. When expanded storage is available, the criteria are a bit more complex:

- The total amount of central and expanded storage should be sufficient to allow all processors to become fully utilized with the desired workload distribution, with little or no page migration to auxiliary storage. Sufficient central storage should be installed (or configured, in a logically partitioned system) to keep the swapping initiated by the Working Set Manager from reaching any workload having marginal performance against its target.

- To exploit the I/O avoidance capabilities of an MVS/ESA system, the amount of expanded storage should be sufficient to accommodate all current and projected needs for ESO (expanded-storage-only) hiperspaces, in addition to

4 On this basis, there is a tradeoff on large water-cooled ES/9000s. On smaller ES/9000s, 9672s, and machines from other vendors, expanded storage is simply carved out of undifferentiated processor storage, and the need for expanded storage is then not an economic choice. The quantity of expanded storage can in such a case be restricted to what is needed for uses such as hiperspaces (including hiperspace LSR buffering), for which central storage is not a substitute. However, it is important to realize that the real storage management algorithms of MVS change when there is *any* expanded storage in the system. It's critical to make sure that enough storage is designated as expanded to deal with any likely demand—or to forego its benefits entirely.

the amount needed as above to prevent unacceptably high page migration. When pages of ESO hiperspaces can be cast out, "enough" means enough expanded storage so that the I/O time required to replace cast-out pages does not threaten service targets.

- For TSO and batch workloads, in which many or most (TSO) address spaces are idle at any given time, relatively less central storage and more expanded storage are needed. Nonswappable subsystem workloads such as CICS have in the past derived little direct benefit from expanded storage other than the reduction of paging delay. In MVS/ESA, placing VSAM buffers and data tables in hiperspace, and using LLA and other VLF data spaces can provide substantial indirect benefits from expanded storage. CICS/ESA (CICS since Version 3) exploits further the ESA environment.

When PR/SM, MLPF, or MDF is active, each logical partition or MDF domain must have adequate real storage, particularly central storage. Even if storage could be reallocated dynamically down to the level of a single megabyte (as it can be on some ES/9000s and newer 9672 models with PR/SM), there must be enough to satisfy the peak demand with some reserve available to support reconfiguration. One benefit of multiple-image options is that any system image may have precisely the amount of storage it needs at the moment. A corollary is that the overall system should always be somewhat overconfigured for storage in order to be responsive to changes in workload needs.

There is no damage done if real storage is slightly in excess of these needs, and there is a loss of apparent CPU capacity if there is insufficient real storage. It would be prudent always to round both central and expanded storage estimates to the next *higher* optional increment size.

Yet another view of storage need is based on the ratio of "megabytes to MIPS." In past days, having 3 megabytes of real storage per MIPS was a common guideline. When the IBM 3084-QX with 128 megabytes became available, the ratio could go up as high as 7 to 1. A 3090-200 was about as powerful as a 3084-QX, so 128 megabytes would seem to have been a normal real storage configuration for a Model 200. The only problem was that the original 3090-200 came with only 64 megabytes of central

storage, so expanded storage had to be added to ensure the best opportunity to load the CPU. Central-storage-intensive workloads such as CICS caused high page movement rates on such systems. That problem was not solved until central storage sizes were increased on the 3090 E models, and increased further in succeeding generations.

10.2.3. Paging subsystem without expanded storage

In an MVS system without expanded storage, the paging subsystem must have sufficient capacity and responsiveness to avoid becoming a system bottleneck and thus keeping workloads from meeting their service targets. Capacity and responsiveness are interrelated. Depending on loading and model-to-model variations, 3390 paging devices can deliver single-page response times of from 15 to 30 milliseconds. When used in a locals-only configuration (without swap data sets), occasional delays waiting for swap I/O (at 42 milliseconds of data transfer time per burst) are traded off for improved average responsiveness based on a larger number of local page data sets.

Estimating system paging rates is covered in King's *Workload Characterization*, cited earlier. Setting up an initial paging configuration based on estimated paging rates is derived from Tom Beretvas's modeling (also presented frequently at conferences and described in IBM Technical Bulletins), supported by extensive measurement data. A basic guideline is that a 9345 (for example) in MVS/ESA can sustain between 29 and 41 page-ins per second. For simplicity, we assume a capacity of 35 page-ins per second per 9345 page data set.[5] We also assume that a 9345 can handle the swap requirements of about 150 TSO users, thus discounting the number of page data sets by one for each 150 TSO users when calculating paging capacity. (See Appendix B for a more detailed view of paging device limits for a variety of device types.)

The page-in rates are precisely the values to be set as upper limits in the compatibility mode OPT parameter RCCPTRT. The lower limit should be 75–80 percent of the upper limit.

> **Example:** A large MVS system has 300 TSO users entering about 15 transactions per second. They each incur about 4 page faults per transaction. There are 5 CICS ad-

5 Only one page data set should be defined per actuator.

dress spaces, each with an acceptable page-in rate of 2 per second, and 1 batch swap per second with 14 page faults per batch swap. Miscellaneous paging is 5 VIO page faults per second and 10 page faults per second from other sources. Total page-in rate is accumulated from these elements:

```
15 X 4 = 60    (TSO)
 5 X 2 = 10    (CICS)
        14     (Batch)
         5     (VIO)
        10     (Other)
        99     (Total)
```

At 35 page-ins per second per 9345 local page data set, we need five local page data sets, three for paging and two more to cover the swap activity of 300 TSO users. The batch swapping is absorbed by this configuration without a need for more data sets. RCCPTRT might be set at (84,105).

Caution: The story is not yet complete. In compatibility mode, TSO and production CICS address spaces *require* storage isolation to function properly in this environment, without expanded storage. (See Appendix B for other considerations on storage isolation in compatibility mode with expanded storage.) Not more than one page data set should be on a path. PLPA and CSA page data sets are needed as well, and a duplex page data set could be used to back up the common-area page data sets to ensure maximum availability. The duplex should be separated from PLPA and CSA by unit, string, storage director, and channel group to the greatest extent possible. If the PLPA and CSA page data sets are on a fault-tolerant device such as RAMAC or another RAID subsystem, the need for a duplex page data set is almost nil.

Page data set size is important too. Assume that swapping requires 120 frames per TSO user, and that 1500–2000 frames are needed per production CICS, 200 frames per concurrent batch job, and about 200 frames per system address space. (More precise estimates appear in King's *Workload Characterization*.) The whole requirement should then be quadrupled, since the benefits of contiguous-slot allocation begin to disappear at about 25 percent slot occupancy per data set and are severely curtailed at 35 percent.

In systems prior to MVS/ESA, a page data set was limited to 65,400 slots, so a number of data sets can be estimated from slot requirements. (MVS/ESA permits full-volume page data sets on

any supported device; the benefit is more in enabling a device to be easily reserved for paging than in creating one huge page data set. The latter choice may be of some use with a large surplus of expanded storage.) The number of required page data sets based on assumed paging rates and that based on slot requirements should be compared and the larger number selected.

Swap Data Sets

Swap data sets are now a footnote in MVS history. They are obsolete, and their use is counterproductive in the vast majority of today's MVS systems. A system with no TSO workload and insignificant batch has no need for swap data sets. A system with nothing *but* TSO and batch also has no need for swap data sets. The only potential candidates for swap data sets are systems with a mixed workload and a significant level of real storage constraint.

Although swap data sets are never truly required, in some unusual configurations they might be of some benefit. Two such far-fetched exceptions are:

- The system is very constrained for real storage, and solid-state devices (SSDs) are being used for paging. Since there is no block paging benefit with solid-state devices, swap data sets on DASD can divert a load that might otherwise cause the solid-state device to overflow paging to DASD. With an adequate solid-state configuration, swap data sets are not necessary. Using the quantities assumed in the example above, 144 additional megabytes of SSD would be needed to support 300 swapped-out users.

- A key workload might have such a pathologically random storage reference pattern that storage isolation is of no help. In the presence of a substantial swappable workload, fast and consistent paging response time requires the use of swap data sets, as well as an isolated cluster of paging volumes or enough solid-state paging capacity to avoid spill.

When swap data sets are used, TSO storage isolation is a must to complete the job of unloading the local page data sets. To minimize the time spent waiting for a subsequent swap-in I/O while holding a partial working set, provide enough swap data sets to accommodate the approximate 90th percentile swap-group size, on separate paths to ensure full I/O concurrency.

Related Considerations

Storage Isolation. As indicated above, storage isolation is an integral part of real storage management in compatibility mode, especially without expanded storage, with or without swap data sets. There are several ways to use this powerful tool:

- **TSO first and second periods:** Specify PWSS=(*sgs*,*), where *sgs* is the desired swap group size. The purpose of this specification is to eliminate up to 60 percent of TSO page faults by including recently referenced pages in the swap-in group. In turn, page data set I/Os are reduced, thus improving demand page response times. The cost (in added data transfer time for swapping) is negligible.

  To estimate the proper value for *sgs*, observe the real storage frame counts for TSO address spaces in the first and second periods at a time of peak usage,[6] using a tool like Candle's OMEGAMON II for MVS or RMF Monitor II. Estimate the smallest value exceeding about 90 percent of the address spaces' frame counts. Round this count up to the next higher multiple of the swap set size for your system. (Local page data sets on 3380s and 9345s have a pseudo-swap set size of 30 slots. Those on 3390s or devices that emulate the 3390 geometry can block pages up to 36 at a time. Swap data sets always use a 12-slot swap set, regardless of device type.)

  Tune *sgs* (and verify that storage isolation is beneficial) by observing execution states for the workload. (OMEGAMON II for MVS is an appropriate tool. RMF Monitor III does not distinguish between swap page-in delay and nonswap private page-in delay.) If PRIVATE PAGE-IN WAIT is significantly greater than SWAP PAGE-IN WAIT, increase *sgs*. If SWAP PAGE-IN WAIT is significantly greater than PRIVATE PAGE-IN WAIT, decrease *sgs*.

- **Production subsystems:** In the single performance period definition for a nonswappable subsystem, specify PWSS=(*min*,*) and PPGRTR=(*ok,high*), where *min* is your estimate of the minimum frame count needed in the address space to provide basic functions without incurring a

6 The frame count at peak activity times is likely to be close to a true working set, due to page stealing.

page fault, *ok* is a tolerable page fault rate, and *high* is the maximum acceptable page fault rate for the address space. The purpose of this specification is to eliminate a large number of page faults by retaining recently referenced pages in the address space. The cost (of limiting page stealing in these address spaces) is seen as added page stealing and consequent page faulting in unprotected address spaces.

- **Unfavored swappable address spaces:** Specify PWSS=(0,*max*), where *max* is the frame count above which you want preferred page stealing to take place. This "negative storage isolation" may be helpful if the preferred workload on the system has an erratic storage reference pattern such that normal storage isolation is ineffective, or if the less preferred workload has an unusually large virtual storage requirement but a much smaller working set over some interval like 1 to 2 seconds. *Do not* use negative storage isolation for *swappable* address spaces in MVS/ESA as of SP 4.2; the Working Set Manager does a more sensitive and effective job of limiting the storage impact of such address spaces at times of constraint.

Logical Swap Controls. The default controls favor logical swapping unless the UIC is consistently less than 20. There is a discussion of various strategies for changing the logical swap controls in Appendix B.

10.2.4. Paging configurations with expanded storage

Expanded storage changes MVS systems significantly. Assuming a sufficient amount of expanded storage, its principal effect is to eliminate direct dependency on page data sets. Enough auxiliary storage slots must still be available to avoid running afoul of auxiliary storage shortage thresholds,[7] but performance becomes a secondary consideration in establishing the paging set-up. In an MVS/ESA system, one full-volume local page data set might seem to be sufficient—assuming there are few occurrences of SVC dumps. On the other hand, normal operation is not guaranteed. When a system dump is being written, a single page data set can be an intolerable bottleneck. Cheryl Watson

7 These values, MCCASMT1 and MCCASMT2, are *not* OPT parameters. They remain as ZAPpable constants and may be found in the MCT control block, mapped by the IRAMCT macro. They are described in Appendix B..

has recommended a minimum of seven page data sets to ensure that there is acceptably low system impact from SVC dumps.[8]

Caution: Make sure that RCCPTRT reflects the capabilities of the actual paging subsystem, with possible derating to reflect some level of page migration. The only time the paging subsystem will be called upon to any great degree is when expanded storage is fully utilized. It is important, therefore, to track expanded storage occupancy and migration rate over time to guard against unwelcome surprises.

The balance of central to expanded storage is reflected in the page movement rate. When this value exceeds about 500 pages per second (in and out) per processor, central storage has become constrained. Other than a central storage capacity upgrade, two moves that can help are to reduce the number of fixed pages and to cut back on logical swapping.

However, the Working Set Manager in MVS/ESA SP 4.2 and later MVS releases detects the increase in unproductive CPU time that indicates central storage constraint and takes steps to protect the system from further degradation. The page movement rate then becomes a somewhat controlled value and thus is not a valid indicator of constraint once the Working Set Manager becomes active. It might be useful from a capacity planning viewpoint to track the growth of page movement rate when there are no swaps due to WSM intervention, regarding the old 500 pages per second per processor guideline as an approximate threshold of concern.

Migration of pages from expanded storage to auxiliary storage is disruptive (requiring use of the CPU and central storage at a time of system stress). Migration that brings the CPU to saturation is a sign of desperately overloaded expanded storage. This condition is very likely to appear in MVS/ESA systems as ESO hiperspaces and high-performance (I/O avoidance) options of DFSMS and of the base control program come into wide use. As ESA use grows and matures, a capacity planner would do well to track migration age over time and yell loudly for help when the highest criteria age threshold (1500 by default for ESCTVIO and ESCTBDS) is crossed consistently.

8 The November/December 1995 issue of *Cheryl Watson's Tuning Letter* contains an extended discussion of paging configurations and the SVC dump impact.

10.2.5. Shared DASD considerations

A given amount of DASD contention has a greater impact in a shared DASD environment than on a single system. Channels, control units, and device controllers have more work to do and are therefore busy longer. GRS or equivalent facilities also work harder with an increased level of cross-system ENQueues. Assessing DASD delay in a shared environment is considerably more difficult than on a single-image system.

When there is a choice of keeping related workloads together or splitting them across systems, the effects of coupling through shared DASD should be a key factor in making the decision. Three developing trends complicate the picture. The ease of creating multiple system images with PR/SM-LPAR, MLPF, or MDF may increase the incidence of shared DASD problems. The sysplex capabilities introduced with MVS/ESA SP 4.1 facilitate clustering multiple systems and system images and perhaps obscure the practical difficulties associated with shared DASD. On the other hand, as system-managed storage in MVS/ESA unfolds, a solution to data sharing may become part of the picture.

Until enterprises make a firm commitment to parallel sysplex and all heavily used access methods are enabled for data sharing, the use of active DASD sharing should be avoided as much as possible. Placing applications on systems so as to maximize data clustering on individual system images and minimize active sharing can help reduce the impact of shared DASD contention.

10.2.6. Use of PLPA and LLA to reduce I/O

The pageable link pack area (PLPA) is a classical MVS tradeoff. Placing modules in PLPA eliminates I/O and program fetch processing by allowing only a single copy of a shareable module to serve all requesters. The price of this convenience is use of virtual storage in commonly addressed storage. As virtual storage constraint below 16 MB increased, taking modules out of PLPA became a necessary activity, especially in MVS/370. In MVS/XA and later in MVS/ESA, the Extended PLPA (above 16 MB) regains the PLPA benefit while easing the virtual storage constraint in the lower addressing range. As more modules become capable of running in EPLPA, ESA systems will again realize the full benefit of sharing reenterable load modules.

When the major benefit of placing a module in PLPA is I/O avoidance rather than module sharing, MVS/ESA makes another option available. The ESA Library Lookaside Area places modules in a data space, making functions provided by such modules available without I/O and without compromising the size of the common area below 16 MB. LLA is slightly slower than PLPA (because Program Fetch is involved), but nonreenterable executable modules are also eligible for inclusion in LLA; they cannot be in PLPA.

Modules that are heavily used by multiple swappable address spaces such as TSO are good PLPA candidates. Those loaded infrequently by nonswappable address spaces could better be placed in LLA. The reduction made possible in [E]PLPA size can then be added to the [extended] private area for all address spaces.

10.2.7. MVS/ESA options to avoid I/O

Library Lookaside

The Library Lookaside Area in ESA provides a way to trade virtual storage in a data space (and its backing expanded or auxiliary storage) for the I/O of program fetch. In MVS/XA, "LLA" stood for *LINKLIST* Lookaside Area, a more limited function that eliminated only directory I/O.

VLF

The ESA LLA function is built on a lower-level function, the Virtual Lookaside Facility (VLF), that can use data spaces to contain whole, or parts of, selected libraries not restricted to load module libraries. In TSO, for instance, ISPF panel libraries and command procedure (CLIST) and REXX (EXEC) libraries can be managed by VLF to trade virtual storage in a data space for I/O.

Data in Virtual

MVS/XA and MVS/ESA both support Data in Virtual, a set of primitive functions enabling an application program to manage a large random-access data aggregate as an object embodied in a VSAM Linear Data Set. Only the referenced pages of such an object are brought into virtual storage. Bytes of the mapped pages can be accessed and changed in normal program execution without regard to the need for updating. On request or at

the time the connection to the DIV object is terminated, only the changed pages are written back to the linear data set.

DIV is somewhat difficult to use because the assembly language primitive functions are not readily available in high-level languages. The potential benefits may be realized eventually as DIV merges with hiperspaces (a related concept) and as subsystems, languages, and application packages exploit the DIV benefits.

Hiperspaces

The DIV experience of limited usage should not have been repeated with hiperspaces. IBM provided a broad spectrum of initial support for hiperspaces in critical subsystems, ranging from direct use for VSAM buffers and CICS data tables to accessible high-level language primitives called *data windowing services.* The lower-level representation of hiperspaces as objects akin to data spaces should have encouraged system programmers and those constructing major application packages to experiment with hiperspaces much more readily than they did with DIV. Hiperspaces also are part of the storage hierarchy of system-managed storage.

In a first step, this option is implemented for PDSEs (Partitioned Data Set—Extended), supported by DFSMS as the eventual replacement for ordinary partitioned data sets. Part of the specification for such a data set is the MSR or millisecond response time requirement, usually used to make decisions relating to dynamic caching. However, a PDSE with a sufficiently low MSR may be placed in a hiperspace.

The Hiperbatch facility made available first in MVS/SP 3.1.3 is another such use of hiperspaces. Based on prespecified eligibility criteria under management of DLF (Data Lookaside Facility), QSAM or sequentially accessed VSAM data sets may be placed in hiperspaces to avoid repeated reading in batch jobstreams. Hiperbatch can provide great benefit in improving the performance of ancient batch applications.

The backlog of new application programs waiting to be designed and written is enormous, and development cycles are long. Perhaps by 2000, major applications designed *ab initio* to make effective use of hiperspaces will realize the full data addressing and performance potential of MVS/ESA.

In the short term, the expected benefits of hiperspaces have not been realized except in IBM-provided exploitation. One possible reason is that there are alternatives to many of the uses of hiperspaces—private area buffering is often more effective than hiperspace buffering. It also would seem that application developers have been too busy looking at portable applications and conversion to client-server to focus on MVS/ESA-specific facilities such as hiperspaces. It's even more certain that developers of commercial applications hold back even more on picking up platform-specific performance enhancers.

10.2.8. Avoidable I/O contention

While many instances of I/O contention are found and corrected in the process of resolving service anomalies in workloads, many, especially those within an application family, can and should be anticipated and prevented before the fact. In designing applications and planning their installation for production, a simple time-line plot can show the planned pattern of use for each data set or data base. In this way, possible sources of conflict can be flagged, leading to an optimal data set placement pattern for the application.

Data set growth and volume fragmentation should be tracked for key applications, particularly those with critical timing windows or real-time response targets and heavy I/O content. Again, while it is possible to detect and correct sudden increases in SEEK delay after overt service problems appear, routine storage management actions should reveal the impending problem before service is affected.

The promise of system-managed storage is that many of these considerations will eventually be handled automatically—if DFSMS is properly instructed with a well-designed class structure. Data placement decisions, made in the initial implementation of DFSMS only at allocation time, might eventually be affected by feedback from performance monitoring during the entire life of data objects.

To the extent that DFSMS does not support such feedback, the storage management task for the installation must continue to include performance assessment of key workloads. Some DASD management tools are available, including Candle Corporation's OMEGAMON II for SMS and Boole & Babbage's DASD Advisor. Their main orientation is to the resource view rather than the

workload view, but the Candle product does provide some application-based grouping capabilities.

It is important for the users of such systems to realize that the object of "DASD tuning" is not merely to create "homogenized" RMF device reports. If every I/O request is processed with the same response time, workloads with modest DASD performance requirements will be too well served, and those that are response-time sensitive will do badly. The perspective is key: DASD tuning (beyond cleaning up fragmentation and contention) is justified only on behalf of workloads.

10.2.9. Path contention

As the number of channels available and installed on systems increases, and as four-path DASD subsystems become commonplace, path-loading considerations fade in significance. Again, system-managed storage could eventually take note of path response times in placing data sets with a fast response time requirement. If system performance depends on multiple I/O paths, it is also important that the necessary paths be monitored for continuous availability.

10.2.10. Using cache control units

Cache controllers provide significant value in systems that are heavily dependent on random-access I/O with a significant re-reference pattern. More than likely, the applications are relatively old and transaction volumes have increased to the point that using cache controllers is the only readily available means of meeting response-time requirements.

The current generation of cache controllers, typified by the IBM 3990 Model 3, the RAMAC subsystems, and EMC's Symmetrix line, has features that reduce the need for laborious planning and validation to decide which data should be cached. However, overt planning is still needed. For caches such as those in the 3990, system-managed storage will ultimately allow a "management by objectives" approach to classifying data according to its performance requirements. With the full power of SMS, that variety of cache will be used to its potential with minimal day-to-day human involvement. In the newer subsystems, the cache is properly characterized as "massive" and all data will be routinely cached.

More advanced features of new cache controllers provide specific targeted benefits for sorting programs and for elements of

designated data base subsystems. Little experience with such facilities has been published to date.

As more applications are reimplemented to exploit data spaces and hiperspaces, the need in such applications for cache devices is likely to be reduced, in exchange for greater dependency on expanded storage. The large caches, available at relatively low cost, will then promote a more general improvement in I/O responsiveness. A more cynical view is that the advent of such large cache subsystems will provide justification for yet more procrastination in updating truly hideous application designs.

10.3. Resource Exhaustion

There comes a time in most systems' lives when upgrades appear to be inevitable. Storage or CPU might be fully utilized, or queueing for CPU or I/O devices has put service targets in jeopardy. To go ahead and order more hardware at this point seems natural. However, the system may not be out of capacity at this time. Reprioritizing the workload may buy valuable time.

10.3.1. Forestalling the inevitable

The CPU is running at close to 100 percent utilization through first shift; is the system out of capacity? If there is excess capacity at off-prime times, and if a portion of the workload has a service requirement that can stand delay, more high-priority work can be added to the system until CPU queueing compromises service targets. Only when that occurs is more capacity needed.

How much planning is needed to keep up with workload growth? If an installation's processors are not at the growth limit for a vendor's product family, most upgrades take less than a day to install and little lead time. In the case of the IBM processor product line, many upgrade paths are possible. Some involve simply adding an additional processor to a frame capable of accepting it. Others may require extensive parts swapping or the addition of a frame or a new power controller.

When a new physical "box" is needed, floor space and environmental considerations can intrude. When a single-sided processor becomes physically partitionable (two-sided) in an upgrade, a large increment of floor space, as well as more installation time, is necessary.

These considerations have been turned upside-down by the availability of high-end CMOS-based systems and the use of parallel sysplex. The cost of an upgrade has been reduced; it is simply a matter of plugging in a few cards which are inexpensive enough to be within the data center manager's discretionary spending authority. When the box (such as a 9672-RY4) is fully populated, another can be bolted alongside it and the processor and memory cards balanced between the old and new boxes. If the configuration is operated as a parallel sysplex, adding another system remains more disruptive than upgrading a processor complex, but far less disruptive than adding a system in a classical shared DASD environment. If the MVS images of the parallel sysplex are clones of each other and share system volumes to the greatest extent possible, adding another system is hardly disruptive at all.

In a large installation with many processor complexes, backup systems can often absorb the workload of a system being upgraded. Shops that haven't yet grown to that level require more care to avoid outages associated with upgrades. Sometimes it is possible to schedule preparatory activity on several successive weekends; more aggregate time is needed for the upgrade, but with less disruption.

10.3.2. Tuning for efficiency

As processor complex capacity becomes exhausted, it is more important than ever to ensure efficiency. Although the constant emphasis in this book is on workload-oriented performance management, at this point our focus switches from tuning workloads to tuning the system. Success based on either approach is usually beneficial at both levels.

Measures that generally help the system and all workloads without negative consequences include:

- making sure that block sizes for sequential and partitioned data sets are as large as the maximums supported on the devices
- making maximum effective use of PLPA to eliminate redundant program loading and to reduce the working-set sizes of address spaces, especially TSO

- making maximum effective use of LLA in MVS/ESA to minimize directory searching and reduce program loading times
- eliminating the use of STEPLIBs for all but carefully restricted test or preproduction applications
- making sure that heavily used or shared data sets are monitored to prevent wasteful seeking across multiple extents. In ESA beginning with SP 3.1.3, use facilities such as Hiperbatch and batch local shared resources to avoid I/O when possible
- in compatibility mode, acting to prevent logical swapping from dominating constrained real storage, forcing unnecessary page stealing
- also in compatibility mode, adjusting the OPT MPL controls to ensure maximum resource utilization without wasteful swap-in delay (when real storage is sufficient) and without requiring manual intervention to control the size of the multiprogramming set

The last two measures are meaningful in systems without expanded storage or in systems where significant workloads have been excluded from expanded storage through the criteria age parameters in the OPT. They also are completely unnecessary in goal mode.

10.4. Resource Contention

When contention analysis is an available tool for diagnosing performance problems, it is easy to fall victim to the Carpenter's Fallacy. Not all performance problems are contention problems. As we've seen, there are some genuine instances of resource exhaustion, and there are all too many examples of poorly designed or implemented applications that create their own efficiency problems. We'll limit our discussion of contention to instances of contention for a named, specific resource such as a data set, volume, or ENQueue resource name.

In this section, we identify some key contention scenarios and suggest ways to avoid them, or to deal with them when they do come up.

10.4.1. Prevention

Suppose that three CICS address spaces (order entry, order fulfillment, and accounts receivable) in an MRO subsystem are set up so that all are in use at the same time, and that they each use the same VSAM customer information file (CIF) for query, verification, and occasional update. Each address space processes a different set of transactions, with different dependent files on the same volume as the CIF being updated, as well as on other volumes. Suppose further that all CICSs have been just barely meeting their response-time targets.

With strong forebodings of impending disaster, we now plan to expand the geographical service area of this system such that the transaction rate can be expected to double. It seems prudent to act now to avoid the problems which seem certain to come. Using or collecting execution-state analysis data for each of the address spaces, we could determine if there is already a concentration of contention-delay states such as "waiting for queued I/O" or "waiting for ENQ."[9] If such states show up, it is almost certain that they will increase dramatically with volume growth.

If I/O contention already exists, it is most likely to appear on the central data volume containing the CIF. Contention within the CICS subsystem may be relieved by splitting the subsidiary files to other volumes; however, the possibility of causing contention with another workload should be a consideration in selecting the target volume. Exercising advanced options such as shared buffering or placing the buffers in a hiperspace in an MVS/ESA environment, especially for the CIF, can help to avoid the I/O that is the source of contention.

10.4.2. Avoidance

How can we avoid creating new contention without leaving resources idle? What we need to find are complementary or compatible workloads. If two workloads need access to the same resources but at different times of the day, there is no harm in allowing the sharing. If they both need the same resources at the same time, they will be in harmful contention unless the sharing can be promoted to a hardware resource level that has ac-

9 Note that this kind of state analysis data is not available from Workload Manger's data collection; a performance monitor that breaks out these delay states across address spaces is needed. Even better, a direct contention monitor shows the identity of the contending address spaces.

ceptably low impact on performance. From best to worst, the levels would include:

- data in DIV or hiperspace, as in Hiperbatch
- data replicated for each using subsystem
- shared buffering (global or local shared resources), including buffering within the coupling facility of a parallel sysplex
- cached, same system or shared in a parallel sysplex
- shared DASD, cached, different systems
- uncached, same system
- shared DASD, uncached, different systems, with global ENQueue management
- shared DASD, uncached, different systems, with no global ENQueue management

10.4.3. Detection in real time

As with any approach to diagnosing a problem, contention analysis starts with identifying a condition that requires investigation or possible action. A real-time monitor such as Candle's OMEGAMON II for MVS can issue a visual alert or display a warning message when there is an ENQueue lock-out or a device not responding or with abnormally high service time. As we've already noted, such warnings are little more than false alarms if the workloads affected by such conditions are not sufficiently important to merit operator attention. A real-time monitor with workload awareness can issue exception messages against the response time or elapsed time of specified critical workloads, most effectively using the history of the particular workload as the basis for the threshold.

When a workload exception appears, the person investigating the problem can use the probing features of the monitoring program to display the execution-state profile of the workload and its direct contention sources. If the monitor does not collect contention data, it may maintain a history of workload activities and resource use that might provide less direct evidence of contention.

Knowing (or strongly suspecting) the identity of the delaying resource and the workload causing the contention, if there is one,

the analyst can now decide whether to take action, and if so, which actions could eliminate or minimize the contention.

10.4.4. Solutions

Let's review what can be done about a contention problem:

- **Tolerate It**—If service is acceptable for the moment, track the execution states of the workload over time, looking for a sudden rise in a contention-delay state, at which time action will be necessary.
- **Accommodate It**—With data set contention, take action to promote the data in contention to a faster device, or to exercise buffering or caching options that can reduce the contention impact. When there is CPU or storage contention, a short-running program causing degradation to a subsystem might be "accelerated" by placing it in a more favorable performance group or service class. Ultimately, application rework may be necessary to cut contention impact to the minimum. Batch jobs that cause CPU contention usually do so by causing a high rate of SRB activity to service I/O completions. The appropriate solution in such a case is to eliminate or minimize the disruptive I/O, usually by increasing block sizes. In the case of batch, the cost of licensing BatchPipes or SmartBatch may be justified if the CPU contention attendant to high I/O rates can be minimized.
- **Avoid by Workload Scheduling**—If the service schedule for one of the workloads in contention can be changed to eliminate the contention, take the opportunity. This approach may be useful if a production batch job interferes with an online service. As the "batch window" shrinks with the growth of extended online service hours, this option becomes difficult to implement.
- **Avoid by Replication**—If data sets in contention are read-only, making multiple copies can eliminate contention. The price paid is a new need to synchronize the copies. This option is unnecessary in a parallel sysplex if the access method is enabled for data sharing.
- **Avoid by Sharing**—Options such as shared buffering and caching serve the same purpose as replication without the administrative burden. If load module libraries are the ob-

jects subject to contention, the use of PLPA or the MVS/ESA LLA can eliminate most contention. Objections to placing unauthorized application modules in the LINKLIST were overcome early in MVS/XA when APF authorization of LINKLIST libraries could be made selective rather than universal. (This change enables eliminating STEPLIBs for production applications.)

- **Avoid by Movement**—If device contention (queued I/O) is the main problem, move data sets to decouple the workloads in contention. This action is effective even if the device contention is self-contention within the same workload.
- **Avoid by Rearrangement**—A CICS workload may have been split across address spaces in MVS/370 because of virtual storage constraint. Now, in MVS/ESA, the same configuration is used because of habit or inertia, even though the constraint is no longer a problem. Reconsolidating an unnecessary MRO or resplitting according to a minimum contention pattern may cut out enough contention to maintain consistent service.
- **Avoid by Rewriting**—The last remedy considered in most installations is to reimplement the application to eliminate sources of contention. A new data base design using options available in relational systems may completely bypass the source of contention.

10.5. Summary

Solving performance problems is often difficult and time consuming. It is also intellectually challenging and very satisfying when successful. Doing the job well is aided greatly by proper preparation and placement in a supportive environment containing these elements:

- suitable tools, especially execution-state and contention monitors
- well-defined service targets
- freedom to make changes in system control parameters
- a performance management job description, evaluation policy, and management structure that recognize cumula-

tive success and support the risk-taking necessary to solve problems caused by the actions or inactions of others

- a capacity-planning and hardware acquisition function in the installation that strikes an optimum balance among projection of past experience, forecasting of future loading, timely acquisition of new technology, serving the performance expectations of customers, and deriving full economic benefit from capital assets

Even though every practitioner develops an individual approach, each such person should take pains to remain open-minded about different approaches. He or she *must* keep current with developments in operating systems and subsystems, hardware, monitoring software, and behavioral studies about human interactions with computer systems. Today's systems do not respond well to yesterday's management techniques.

And that's not all! It's also necessary to know a few more practical things: what the principal applications do, and how the customers view the service of which those in the data center are so proud.

When the performance analyst has proper preparation, knowledge, good tools, support, operational discretion, and an open mind, no performance problem should long remain a mystery. Whether it can be fixed may depend on other considerations. We'll look at some of them in the next chapter.

10.6. Chapter Questions

1. What is the performance management methodology in your installation? Is it proactive or reactive, workload-based or resource-based, after-the-fact or real-time? Is application performance part of the system performance charter?
2. Based on the material in this chapter, can you revise your answer to question 1 with confidence? What new techniques can you add to your methodology?
3. Examine your paging configuration based on the example discussed in this chapter. If you are using swap data sets, see if you can state the justification for doing so. Prepare a plan to revise the paging configuration as suggested

herein, and to measure system and workload performance before and after the change. Try it.

4. Look at the system tuning suggestions made in this chapter. Evaluate your system against each of the areas covered. Create a plan to implement any of the suggestions that appear productive for your system.
5. Look at your principal application workloads. For each of them, evaluate two propositions.

   a. "I/O avoidance will help the application."

   b. "I/O avoidance will help other applications or the system."

   Create an appropriate action plan, but read Chapter 11 first.

Chapter

11

Application Performance Management

In many installations, the last area to be influenced by the data center's performance staff is the set of production application programs that to a great extent justify the existence of the data center. Those responsible for such programs are in departments with names like *Management Information Systems* or *Applications Development*.

Typically, these departments are separate from the data center organization through the director or vice-president level of management, so sharing of common goals and cooperation do not arise naturally out of organizational proximity.

11.1. Open-Shop Workloads

As urgently as production application programs need performance analysis and optimization, nonproduction or "open shop" batch and TSO may need such attention even more. A machine whose basic workload is administrative and nontechnical must be managed very carefully if it is also to accommodate such applications as engineering design, scientific data reduction and analysis, or econometric modeling.

Typically, those who create and run such programs never finish them—they never really become production programs. They are tweaked here and there, adjusted to different cases of interest, or run with more or less data or for more or fewer iterations. More than most "commercial" programs, these *numerically in-*

tensive computing (NIC) applications can be disruptive to systems. This is especially true if they are written by professional engineers, scientists, or statisticians who happen to be amateur programmers. (A common end-user response to news of a hardware upgrade or operating system improvement in handling of NIC programs is "Good, now we can run bigger cases.")

CPU utilization is not the problem; it's easy to manage dispatching priority properly. Rather, numerically intensive programs can put tremendous stress on the processor storage resource when they exhibit poorly planned storage reference patterns. The techniques mentioned under "virtual storage" later in this chapter can be tried (or suggested to the program owners). Programs that are just big, even if well written, are inherently disruptive. The blocked paging enhancements and the rest of Working Set Management introduced with MVS/ESA SP 4.2 showed dramatic improvements in mixed-workload systems. The extensions to Working Set Management and the introduction of Workload Manager goal mode in MVS/ESA Version 5 make it even more likely that these workloads can be effectively tamed.

11.2. Application Performance Vulnerabilities

Almost any performance management action taken by the data center staff can be undone by implementation or installation choices for applications. It is unlikely that anyone gets up one fine day and says, "I'm going to mess up the CICS production system." What's more likely is something like one of the following:

- A "portable" packaged application is installed without examining the data definitions and JCL and adapting them to installation standards.
- A new program has been written, compiled, and installed with unchanged default parameters for I/O and storage use—defaults that may last have been examined in 1978.
- An unoptimized prototype has been placed in production.
- A new production program has been "promoted" without removing debugging code (completely).
- A new version of a production jobstream has been installed *via* STEPLIB.

- Input or transaction volume has grown to the extent that formerly efficient code has become inefficient today.
- An obsolete policy of volume ownership has forced inappropriate DASD volume sharing, creating inescapable contention among related applications.
- A similarly obsolete policy of application affinities has created a CICS MRO configuration that makes it impossible for goal mode to properly manage the performance of important transactions without giving undeserving, less important transactions a free ride.
- An upgraded system environment has failed to deliver its potential throughput or performance improvement because the new facilities cannot be used by old programs without change.
- An upgrade was ordered and installed without sufficient analysis to expose and deal with secondary constraints that remained hidden until a primary constraint was relieved. Latent demand associated with an application area may serve to expose the weakness in the new configuration.

It is clear that these circumstances involve no malicious intent; some of them represent lack of care in areas that don't appear (to the application programmer) to be important. Most, however, are simple instances of the application developers not having "system" knowledge that the systems or performance staff thinks of as self-evident.

A system programmer can hardly avoid staying current with hardware or operating system developments. Application programmers are much more attuned to the target areas of their applications and do not normally receive detailed information on new system developments or capabilities. Organizational separation contributes to the lack of information flow.

11.2.1. Vendor application packages

If an applications group depends heavily on vendor-supplied application packages, it is almost inevitable that those packages have attributes that contribute strongly to inefficiency:

- They are designed to be portable across many levels of system environment, so that packaging, data set and data

base definitions, and use of system services tend to be at a "least common denominator" level.

- They are implemented to satisfy functional needs primarily, and are not particularly optimized for performance.
- They are designed to be reliable and therefore are not implemented to assume the risk of using relatively new and untried features.
- They are designed for generality and therefore are not optimal for a particular installation's needs.

11.2.2. Turf wars

Data center performance analysts or specialists are usually staff people in their own departments. They are even more organizationally remote from applications developers. To be effective in getting the applications staff to change JCL, options, schedules, or even *code*, the performance person cannot rely on organizational authority. A person in such a position must cultivate the art of using *influence*—getting others to take action because they perceive it to be in *their* best interest, not because someone in authority said so.

Effectiveness in building and using influence is based on trust, credibility, and respect. Techniques that help to build and rehabilitate a data center's reputation with its customers in general are just as necessary and effective when trying to get only a few of them to accept recommended changes.

11.3. Internal Inefficiency

It's hard to tell when an application program is *effective*. Nothing a performance monitor can measure can tell if the program is solving the right business problem.

Efficiency is another matter. An inefficient program stands out as *different* in its resource utilization pattern, and more so when delay states in the execution profiles are examined as well. Such programs often corrupt other work in the system with avoidable contention. In the next few pages, we'll look at factors that contribute to inefficiency, adding recommendations for improvement when the cure is not evident from stating the cause.

11.3.1. CPU delay

Contention for the CPU resource is managed through the dispatching priority mechanism as administered by the MVS dispatcher. With Workload Manager goal mode, it's no longer necessary to go through the dreary exercise of laying out a dispatching priority order. The dynamic interaction of workloads, goals, and levels of importance determine the current order of dispatching priorities.

Systematic granting of access to the CPU in accordance with the priority scheme is upset by use of the CPU at high priority on behalf of operating system functions. Usually the service burst on behalf of paging or I/O is short, so low activity rates are not harmful. As activity rates increase, the likelihood of the CPU being busy when an important workload is ready for CPU service increases, lengthening response time by incurring CPU delay.

The trend toward multiengine configurations has reduced the significance of "waiting for CPU" as a cause of delay in key workloads. The trend toward logical partitioning of processor complexes works the other way, making the CPUs appear less powerful and sometimes setting an absolute limit on the CPU fraction allocated to a partition. Regardless of the configuration, if the CPU resource is anything less than overloaded, CPU delay can be directed away from time-sensitive key workloads through tuning efforts.

The absolute magnitude of typical CPU delay is much smaller than that of I/O delay or real storage delay. It might be considered a sign of good performance management when the CPU has become the "last bottleneck."

11.3.2. CPU use

If CPU delay is not significant in the total set of execution states, simply using the CPU might be. There is no limit to the inefficiency that can be put in a program. Ignoring the full power of the System/390 instruction set, choosing the wrong data format, leaving invariant code in inner loops, using interpretive techniques when a compiled alternative exists, choosing an inefficient algorithm for an iterative procedure—all these and more are seen over and over by performance analysts trying to account for excessive CPU consumption, especially in a new or changed application.

Specifics include:

- using "roll-your-own" sorting routines instead of using efficient standard sort algorithms or advanced data base techniques that reduce the need for sorting.

- ignoring standard optimization techniques, such as removing invariant code from inner loops, using early-decision logic optimized to path frequency, and "unrolling" function calls within inner loops. Although advanced compilers can do some of these things automatically, "efficient" assembly language programs must be hand-optimized.

  The sheer volume of assembly language source code, the difficulty of imposing structure on existing unstructured code, and the considerable egos of many assembly language programmers make it unlikely that such techniques will be applied to many programs written in that language. Growing popularity of C and C++ as all-purpose programming languages make attention to compiler optimization (and the programming habits that can negate optimization) all-important. A plus for C and C++ is that compilers for those languages in the MVS/ESA environment tend to be of recent design; a minus is that the compiler itself may suffer from the design compromises of portability.

- failing to use efficient in-line-code alternatives to expensive CALLs on library routines or system services.

- failing to recognize vectorizable algorithms and to use vector processors when available.

- failing to recognize program sections that can run in parallel and to exploit multiengine configurations.

- using inefficient and disruptive techniques such as scheduling SRBs in order to communicate across address spaces, instead of using cross-memory services.

- failing to follow through to production on programs developed according to a modular discipline. When such programs are promoted to a production status, linkage code that externalizes the modular design should be removed; the program should be "collapsed" to an "encapsulated" state once it is ready for production.

Using object-oriented languages and design techniques has the potential to produce efficient programs, but only if the initial design models are created with efficiency as a basic requirement. Those popularizing this technology tend to emphasize reliability and reusability of code, leading to inexpensive and rapid application development. Performance is not on their list, even though it could be if the cost-oriented focus could be broadened.

11.3.3. Virtual storage

As the migration to MVS/ESA proceeds, virtual storage limitations will become relics of the past as the full power of data spaces and hiperspaces is exploited. However, the availability of essentially unlimited virtual storage increases the likelihood that some improper programming practices will cause problems in the management of real storage, as we'll see next.

11.3.4. Real (central) storage

In a virtual storage operating system, real storage is a hidden resource. Application programmers in particular have no way of explicitly manipulating central storage. The use of central storage is governed indirectly, by the interactions of workloads and by coding practices within programs.

The demand for central storage is difficult to estimate. Even usage is hard to understand. For instance, a job may take 130 frames of central storage when the system is lightly loaded, but only 75 frames when storage is heavily in demand. Which is the correct characteristic number? What amount, if any, should the user be charged for?

An answer can be found in the concept of *working set*, introduced by Peter Denning early in the history of virtual storage environments. An operational definition of working set in today's MVS terms is "the minimum number of central storage frames needed by an address space to accomplish its purpose without incurring excessive paging delay."

This definition is very flexible. If the paging mechanism is faster, the working set can be smaller. If the "acceptable" level of paging delay can be increased, the working set can be smaller.

A program's working set may increase if the data to be most frequently used is spread out over an unnecessarily large number of pages or if a large data area is sequentially searched for a

data item, rather than finding it directly using an index or hashing mechanism. The efficient opposites of these inefficient practices are called *locality of reference* and *validity of reference*, respectively.

Program design principles for good real storage performance were examined within IBM when virtual storage operating systems were first being explored. A product of that research was a set of three simple rules:

- **Design for good locality of reference.** Ensure that data to be used together or closely in time is close together (on the same page or on a minimal set of pages) in virtual storage. Equally, ensure that data that will not be needed together is not grouped or ordered according to a little-used property so as to rule out locality for data that is related by more frequent use. More virtual storage may be needed overall, but less central storage should be needed at any given time.

- **Design for good validity of reference.** Ensure that each data reference moves the process substantially closer to the desired result, rather than simply ruling out an uninteresting data element or "touching" a data element when no immediate use will be made of it. A search through a linked list that fits on one page may be acceptable; one that wanders over 200 pages may on average cause up to 100 page faults before a hit is found. Creating an index or hash table to map such a data structure can concentrate the searching on one or two active pages and then might cause one page fault when the correct locality of the desired data element is found.

  A second type of validity violation occurs when a large data area is initialized by preloading data values in anticipation of possible future need for some subset of the data elements.

- **Design for minimum working-set size.** In some cases, this means minimizing virtual storage occupancy, as when compact forms of data storage are chosen over uncompacted forms. In other cases, there is little effect on virtual size, for instance from ensuring page alignment of control sections and related data areas. In still other cases, virtual

storage use will be increased, as when a common data table is replicated for use in several different contexts. The main contributor to minimum working-set size is locality of reference, discussed above.

These three guidelines help programmers create applications that make effective use of the central storage resource. It is important for those programmers to "unlearn" one dangerous habit: regarding addressable storage (virtual storage) as the expensive resource. A spread-out program with good locality of reference in each of its parts may be a much more efficient user of real storage than one that is laboriously woven together in a minimum of virtual storage. Programs do not wait for virtual storage, and no one pays for it—it's real storage that's the real resource that has cost and performance.

The principles of good design for a virtual storage environment are violated frequently. Some examples:

- **Wrong-way tables:** When all of the data for an application subsystem fits in a few pages of storage, the layout of data tables is not a matter of great concern. Suppose, however, that the volume handled by the subsystem has expanded to the point where data tables take up 500 pages. If the tables are organized functionally (all "A's" followed by all "B's," etc.) and referenced randomly by a common index value, a particular transaction or thread may touch 60 or 70 pages. In violating "locality of reference," "minimum working-set size" is violated as well. If, on the other hand, the data tables are "turned sideways" so that each transaction's or user's data is contiguous, the storage reference pattern is significantly improved. Correcting locality will reduce the working set.

- **Inefficient search logic:** In pioneering programs in the System/360 era like the FORTRAN H compiler, chained searching of large dictionary structures was a state-of-the-art technique. In a static real memory environment, chained searching can be very efficient, compared to simple linear searching. In a virtual storage environment, however, it's poison.

  Chained searching is the classic violation of "validity of reference." Binary searches or other hashing algorithms are

faster for large tables and use less real storage. Linear searches remain very efficient for small tables.

- **Wrong-way arrays:** A classical problem in translating programs from FORTRAN to PL/I was the need to understand the difference in array storage layout.[1] The original basis for this need was to deal with esoteric considerations like converting EQUIVALENCE in FORTRAN to overlay defining in PL/I.

  A much more practical need exists in virtual storage systems. Regardless of the programming language, an array has a natural storage order. If a program uses the elements of that array in some order other than its natural order, its working set will be larger than necessary. As in the case of wrong-way tables, changing the order of array references (or of array definition) to make the most frequent reference pattern the natural array storage order will serve to minimize working-set size. Especially with vector processors, establishing correct array reference order is a "magic" optimization technique with no negative consequences.

To localize the effects of programs with inefficient storage reference patterns, the technique of *negative storage isolation* described in Appendix B may be used in compatibility mode, but only for nonswappable workloads. The MVS/ESA Working Set Manager does the job much better for swappable address spaces, and goal mode manages storage protection dynamically in the context of managing workload performance to goal.

Of course, negative storage isolation must be fitted to the problem at hand. A program calculating eigenvalues without the use of sparse matrix techniques may have a 1-second working set of 10,000 pages and a 10-second working set of 60,000 pages. Specifying an ordinary level of negative storage isolation like PWSS=(0,2000) might cause such a program to page-fault continuously and never complete. A more appropriate storage isolation of PWSS=(0,10500) for such an address space might cause a modest paging-delay elongation of the job while freeing the rest of the system from page stealing.

1 FORTRAN arrays are stored in column-major order, while PL/I arrays are in row-major order.

11.3.5. I/O

The final hardware resource to consider is input and output. We restrict the discussion to the I/O to and from the storage subsystem as opposed to that between a program and the human(s) it serves. As important as that interface is, it accounts for relatively little resource consumption. I/O involving DASD and tape is the principal component of elapsed time in most batch jobs and a major contributor to response time in interactive applications and subsystems.

All of our accumulated data processing experience has been reinforcing another wrong message: "I/O is cheap. Save CPU and storage by doing I/O." There was a time when this might have been good advice. Certainly before the advent of virtual storage there was no defensible way to read a file as small as a megabyte into working storage, change what needed to be changed, and write back only the changed parts. Doing that for a file encompassing tens or hundreds of megabytes was out of the question. Most application programs were set up to process such files sequentially. If a master file of insurance policies was to be updated, the updates would be sorted in the same order as the master, and master and update would be sequentially merged to form the new master file. If a million records had to be read and a million written to update 2000 records, so be it. Minimum CPU and minimum storage were used to do the job.

In today's world, however, we recognize the wasteful nature of such processing. It makes much more sense to organize that data as a data base, inserting or changing records in response to the individual transactions as they arrive, with I/O only to the affected parts of the file (data base). Such an approach also keeps the data available to multiple users instead of locking out the whole file, as will happen during a sequential update.

With MVS/ESA, there are new techniques available for avoiding I/O. Data accessed through the channel subsystem moves slowly compared to data moving within levels L1 through L3 of the storage hierarchy described in Chapter 3. The fastest channel-attached solid-state device takes 15 times longer to move a page of data to central storage than moving the same data from expanded storage. When access time (to find the data) and channel-control unit protocol times are added in, the ratio can approach 50 to 1.

To avoid having I/O time become a significant bottleneck in application response time, consider these new rules for the '90s:

- Recognize that passing a file across a program more than once is usually wasteful and unnecessary.
- Think of DASD as "backing store" for virtual storage or hiperspace. Data manipulation should be done in storage rather than as I/O.
- Do the minimum necessary I/O to bring only required data into processor storage.
- Do I/O to put away changed data to keep it safe (commit) and when finished with it.
- Don't use DASD as a sequential medium unless all of the data in the file is needed in every execution of the program, or the file is so small that the overhead of locating the desired data item is greater than that of reading the file.
- Even if the data is organized for direct or keyed access, avoid other DASD I/O to the greatest extent possible by using hiperspace buffering or VSAM local shared resources.
- Reserve sequential file processing for uncritical applications—until they can be changed.

A collective name for the set of I/O avoidance techniques in MVS/ESA is *data-in-memory.*

Unavoidable I/O

As important as I/O avoidance is, much I/O activity remains in MVS/ESA systems. Here are some suggestions for minimizing the impact:

Sequential Data Sets: Experts, including Siebo Friesenborg of IBM and Dr. H. W. (Barry) Merrill of Merrill Consultants, have been saying for years that small block sizes for sequential data sets are inefficient. Consider the consequences of small block sizes:

- Inefficient device space utilization: Especially on IBM 3380s and other devices with a similar underlying track format, data bytes per track decline dramatically with small block size.[2]

2 Chapter 4 has a full discussion of this point.

- Reduced data transfer rate: This is a direct consequence of reduced track capacity and fixed rotational speed, and leads to elongated response time per logical record.
- Disruptive CPU overhead: The high I/O rate associated with small block sizes causes at least two state switches per I/O, with increased CPU time disabled for interrupts and under lock. Every other address space in the system pays the price of increased CPU delay.
- Increased storage occupancy: With lower effective data transfer speed, programs take longer to run, increasing central storage use.

Merrill has recommended consistent use of the maximum block size supported on the device. With such block sizes, all four of the factors cited above take on their most efficient values.

When sequential data sets are managed with generous buffering and optimal (maximum) block sizes, data transfer is usually overlapped with processing, so the slow speed of DASD does not significantly influence processing speeds.

Randomly Accessed Data: Randomly accessed data sets and data bases present a different challenge. In a transaction-processing environment, there is no way for the transaction to "do something else" while the I/O necessary to complete a transaction proceeds. When I/O predominates in such an application or subsystem, performance improvement can come only from avoiding I/O or making it faster.

I/O can be avoided by application design or redesign. Many applications fail to make full use of available virtual storage, performing I/O to repeatedly read small tables or other data areas instead of bringing them into virtual storage once and letting SRM and RSM manage the data. Such practices are common in "portable" applications that must run in several different operating system environments of differing storage architectures and sizes. When larger data aggregates (external data sets or data bases) are involved, basic design principles like validity of reference are just as effective in avoiding explicit I/O as they are in avoiding paging.

If there is significant re-reference activity in the aggregate of data used by an application, the use of cache controllers can bring about significant I/O response-time improvement. As

cache devices mature and grow in maximum cache sizes, restrictive cache planning considerations ease; often whole DASD subsystems are cached.

If data has a very concentrated reference pattern, and in applications with extremely stringent response-time targets, pure solid-state devices offer ultimate response time for channel-attached DASD. As SSDs have become obsolete for improving the performance of paging subsystems, they have found new uses as conventional I/O devices to solve special performance problems.

More extensive redesign of applications will bring about further improvements as the applications move fully into the MVS/ESA environment, making effective use of data spaces (overcoming virtual storage limitations), hiperspaces (for scratch pad and large temporary data aggregates), and Data in Virtual (for large permanent data aggregates with sparse reference patterns).

IBM's DB2 relational data base system, and applications built on it, can take full advantage of many of ESA's new storage options.

11.4. Internal Contention

In the previous section we looked at efficiency of application programs and systems independent of contention. When there is contention, we must determine whether it is caused externally or internally. External contention is found by methods described in Chapter 11. Tools such as Impact Analysis, found in some products of Candle Corporation, will also reveal self-contention, usually for I/O devices.

Self-contention is often found in applications or subsystems when multiple data sets are on the same device. A frequent cause of such data set placement is a DASD management policy based on "volume ownership." The trend to system-managed storage substitutes a much less restrictive volume pooling concept and allows the specification of desired performance characteristics as part of data set definition. As such subsystems evolve to include workload performance data in their criteria for data placement and continued residence, I/O device contention should migrate to the least important workloads on the system, subject to the capabilities of the configuration.

Another cause of self-contention, this time for the CPU, is a program organized with multiple tasks or multiple communicating address spaces running in a system without enough CPUs to sup-

port the level of simultaneity designed into the program. There is no contention if the internal multiprogramming is of functions not usually invoked concurrently. When there is active internal CPU contention, most performance monitors can't reveal it, since execution states are determined for the whole address space rather than for tasks, and the "waiting for CPU" execution state is subordinate to "using CPU."

11.5. Why Applications Run Poorly

Clinging to outdated assumptions about relative cost of resources and being insufficiently aware of the operating system's mechanisms lead to inefficiency and create problem applications. If the misconceptions about values are institutionalized in a chargeback system, more than technical rethinking will be needed to overcome the damage already done. We'll now examine some factors, both technical and "technopolitical," that can lead to unsatisfactory service or squandering of resources.

11.5.1. "The dead hand of the past"

Precedent and tradition are good guides in many areas of life. However, they are not a suitable basis for application design. To do something "the way it's always been done" is to guarantee that past performance sets an upper limit on future performance. What was appropriate for yesterday's technology could be not the best choice for today's. Even though it is unlikely, it would be a good investment to consider re-engineering applications that predate the availability of MVS/ESA.

11.5.2. Choosing the wrong tools

Depending on precedent is understandable. It's comfortable. When precedent is not a factor, the choice of the wrong tool or technique is more likely to be a consequence of limited experience—simply not knowing. A commonly encountered area for such bad choices is the selection of a sorting algorithm. Many such algorithms exist, and their performance characteristics are quantified in standard reference books, usually in terms of a fixed initialization overhead plus a function defining the processing cost per record at different volumes. Depending on a familiar sorting routine that is economical for 5000 records may be disastrous when the application volume scales up to 50,000 records at a time.

In the specific case of sorting, the design question that ought to be asked is whether sorting is necessary at all. When the records are large and the keys small, it might be preferable to build as many indexes over the natural sequence set of the data as there are orders of access. Especially if a data-in-memory approach is being used, the order of the sequence set should be chosen to maximize locality of reference.

Courses in algorithm design in today's Computer Science curricula focus on tradeoffs—for instance the tradeoff between execution time and space, both in processor storage and in secondary storage. Perhaps application designers could use a refresher course to learn how to apply such criteria.

11.5.3. Distrust of new goodies

Some of IBM's early MVS products looked good on paper, but failed in the marketplace, mostly because of unacceptably poor performance. A case in point was VSPC, a subsystem designed to offer personal computing, with APL and other languages, to multiple users. It was supposed to be a high-performance alternative to TSO at a time when TSO had a reputation for poor performance. VSPC was designed to be "bullet-proof" in that it made minimal use of MVS's interfaces. In particular, demand paging was not trusted. Instead, VSPC made heavy use of forced page-outs and requested page-ins.

One peril of such a design is that the designers must guess right, not only at the time of conceiving the design but for the full lifetime of the product. Another difficulty is that such a design cannot take advantage of improvements in the operating system in the areas that were perceived as shaky at the time of design decisions.

Fear of the unknown has led application vendors to avoid supporting 31-bit addressing, to stick to private paging and swapping schemes, and to persist in rereading small data sets in applications artificially constrained to design points better suited to first-generation personal computers than to today's MVS systems.[3]

The I/O avoidance possibilities in MVS/ESA can bring about revolutionary improvements in application program perfor-

3 This is not a far-fetched contrast. The personal computer on which this work was prepared has 48 megabytes of RAM and 6.2 gigabytes of DASD. The CPU is an Intel® Pentium running at a clock speed of 150 megahertz—equivalent to almost 10 mainframe MIPS.

mance, capacity, functional completeness, and operational flexibility. The advent of distributed relational data base systems used in client-server application architectures can concentrate volatile data near the point of use, thus reducing workload and time pressure at central sites. Such benefits are available, but only if the new capabilities are learned and accepted by designers of new applications.

11.5.4. Misuse of new goodies

If avoiding an unfamiliar new feature can result in delaying its benefits to a community of users, rushing headlong into its use may give it such a bad reputation that it may never again be used. Early experience with a feature as old as Virtual Input/Output (VIO) was in many cases so negative that VIO was summarily excluded from many installations. VIO had controls, but only indirect ones. Now, with system-managed storage in DFSMS, VIO can be controlled precisely, and the implementation of VIO now makes use of expanded storage when it is otherwise unused. Use of expanded storage for VIO can be controlled in compatibility mode at the individual workload level using the criteria table entries described in Appendix B. However, it's a safe bet that the installations that were burned by VIO in the early days will not quickly open it up with DFSMS and expanded storage.

IBM has made a considerable effort, with specifically targeted publications, to get application programmers to use the extended addressing capabilities of MVS/ESA directly in new programs. The bewildering array of extended addressing options [data spaces, hiperspaces, Data in Virtual (DIV), DIV hiperspaces, data windowing services], with each in several variations, makes it important to study the choices and the nuances of use before electing to use one of these techniques.

There is no doubt that these functions work. The operating system and its subsystems make use of them with great success. They are available to be used by anyone who takes the time to learn how to do so. In contrast, the learning curve is much less steep using the integrated development environments (IDEs) supplied with many PC program development tools. They promote object-oriented techniques, and include powerful debugging and tracing tools. The emerging integrated client-server development tools in OS/390 should have many of the same characteristics.

11.5.5. Perils of portability

Another force acting to retard exploitation of new operating system capabilities is the desire to create "portable applications." The dream is as old as the first successful high-level language, FORTRAN—over 30 years. If an application were to be written in that subset of FORTRAN that is portable across many different hardware architectures, system implementations, and operating systems, it would have to be written only once! Think of the profits!

Unfortunately, the lowest common denominator of a supposedly standard and machine-independent language is very low indeed. Any feature that is an implementor's extension goes first. Features that depend on the word size, character code, or collating sequence also must go. What is left might be reasonably useful for pure computational activity near the core of the language, but any communication with the outside world is severely crippled.

Thus it was that over 15 years ago, a popular finite-element modeling package included a program in FORTRAN to sort its control-card-image input "deck" to prepare for execution of certain procedures. The sorting was done one character at a time, with no assumptions about the collating sequence. Each character occupied a full-word FORTRAN variable. That sort step was so slow that it was the longest-running job step in jobs that lasted less than 20 minutes on an IBM 3033MP. Simply replacing that program with a skeleton PL/I program that invoked the standard SORT/MERGE package reduced the step run times to insignificance.

The vendor of the package did not care about the performance of the sort step, because most users of the package applied it to very large model cases that ran in hours, not minutes. The package ran (and still runs) on many different computers. The only victims of portability were the engineers who tried to run small models and incurred disproportionately large data processing charges. According to recent reports, that control statement sorting routine remains today as it was 10 years ago.

A package designed for portability or broad applicability has inevitable built-in compromises:

- little or no use of advanced system services

- minimal use of advanced data definition choices
- one-size-fits-all algorithms, even in sensitive areas like sorting
- "least common denominator" parameters such as block size
- volume assumptions that may be very wrong for you
- limited tuning or customization opportunities
- an "easy install" procedure that makes it necessary to dig for what customizing options there might be

New waves of portability are rolling over us today. IBM's Systems Application Architecture (SAA), SAP AG's SAP R/3, and offerings from other vendors point to a future in which many applications will run very much alike, but not necessarily equally well, on machines of dissimilar architecture and widely differing size.

What can the application programmer do to avoid mediocrity in applications he or she installs or maintains? The first need is to be involved *early* in the procurement cycle, before a purchase decision acquires unstoppable momentum. It's good to have a list of stringent performance requirements, including specific needs for tuning guidance in the target environment and the set of operating system features to be supported. No matter how portable an application might be, the vendor is likely to make accommodations for specific operating environments in the finished package if it is clear that sales depend on a proper fit.

11.5.6. Prototyping tools

Many installations have what might be regarded as prototyping tools—program language packages that enhance productivity of the end-user-as-programmer at the expense of code efficiency or system resource use. Fourth-generation languages, "expert system" packages, languages like APL and interpreted Basic, QMF, even TSO CLISTs are in this category. The proper use of such packages is to empower those who are not professional programmers to solve *occasional* problems using the computer.

When "occasional" slides over to "frequent" and then to "continual," it is time to consider whether the wrong tool is being used. A CLIST running at 2 percent of the efficiency of a compiled program may be acceptable if used once a day—but if two hundred TSO users have copies of the "neat gadget" and use it ten times a

day each, the aggregate cost (although very hard to spot) becomes substantial. With a working prototype satisfying the functional requirement, any capable application programmer can write a program to do the job at far less cost. At the very least, it could be converted to a REXX EXEC with a significant performance gain. Opportunities must be sought out, because these sources of inefficiency tend to be well hidden.

11.5.7. Faster is not necessarily better

The development and wide availability of tools for Computer-Aided Software Engineering (CASE) leads to a new kind of problem. When everyone recognized that a programming development project was large, much time and effort were put into design. The passage of time allowed early design flaws to be caught and corrected. By the time coding began, everyone on the project had internalized the concept of the project and was able to work well toward a common goal.

With CASE making coding easy, there is a strong temptation to rush into coding before the design has matured and before all of the developers are "on board." The unspoken principle seems to be: "There's not enough time to do it right, but always enough time to do it over."

In an environment where coding is easy, there is more reason than ever to proceed slowly during the conceptualization phase of a project. The design can be made much more nearly perfect before coding begins.

11.5.8. Leaving the stitches in

After the coding is done and the application system is built, it may turn out to be a poor performer. One possible cause is that the main concern in early building is ease of debugging, rather than best performance. As unlikely as it sounds, production applications are often delivered with debugging code left in (although it might be disabled), or the modularity that was a good idea at the coding level shows up as operational inefficiency. Instruction-trace studies of MVS systems in the past have shown heavy biases in favor of linkage instructions and GETMAIN SVCs. Logical and computational function take a back seat to simply getting around the address space and acquiring storage to do useful work.

We might call this implementation practice "leaving the stitches in." There is no inherent conflict between enjoying the implementation benefits of modularity from design to debug, yet achieving efficient seamlessness in production. The goal of eliminating the stitches should be a mandatory part of the design and implementation process.

11.6. Application Design Choices

When one has the luxury of designing a new application from the start, it's possible to make design choices in the interest of good performance. Some suggestions follow.

11.6.1. Get to the essence

As a designer reviews the high-level objectives for an application, it can be useful to "step back" and reflect on its overall purpose. Is the primary focus to provide information to online users? Is it to produce reports for management according to a prescribed format? What data is to be accepted? Stored? Combined? Changed? How?

Keeping the high-level concept in mind may help the designer to make correct decisions affecting resource utilization. Recognizing the most frequent processing path might help her or him to decide what the sequence-set order of a data set should be.

11.6.2. Design for good real storage behavior

Follow the three rules cited above to make locality of reference, validity of reference, and minimum working-set size key guidelines for the design. Here are some supporting suggestions:

- Make sure that the natural data order coincides with the most frequent processing order.
- Balance the resource cost of indexing against the costs of sorting and searching.
- Revisit locality of reference after structured design is done.
- Balance the cost of data area initialization (a major violation of validity of reference) against whatever increase not initializing (or initializing as needed) might bring in per-item processing cost.

11.6.3. Use top-down design

Postpone the flood of detail in the design stage by employing well-known techniques for delaying low-level decisions. Ensure that the design is both effective and correct by using two related techniques:

- *walkthroughs* to validate high-level design
- *inspections* to validate low-level design and remove defects or errors in carrying the concept down to the level of design that determines what is to be coded

11.6.4. Make performance part of the design criteria

For many years, Dr. Connie U. Smith has been teaching and writing about a discipline called Software Performance Engineering. Her work is well-represented in the *Proceedings* of CMG conferences over the years. The ideas suggested below are in the spirit of Dr. Smith's work:

- Choose algorithms based on largest expected volumes.
- Choose data layouts to optimize fully loaded performance.
- Anticipate full operating system functionality. When a new release of the operating system can be expected on-site before the completion of a project, learn about the "new goodies" and include their use in the design, especially if the payoff is substantial.
- Analyze data reference patterns to avoid contention. If I/O avoidance techniques are used aggressively, the common problem of I/O contention, both intra-application and inter-application, can be minimized. If such techniques are not used (for instance, because of a temporary shortage of expanded storage), I/O pattern analysis in advance can save frenzied tuning effort after installation.
- Optimize inner loops at the design level. Recognize which routines will be most active in the normal running of the application and take special pains to avoid placing invariant computations in such areas, and to avoid performing I/O from within those active spots. If only sparing use can be made of I/O avoidance techniques, focus on the hot spots.
- Always keep the cost of I/O in mind.

11.7. Ideas for Application Implementation

11.7.1. Use proven techniques

There is a host of well-known implementation techniques (many of them rooted in design methodology) that have produced outstanding results in eliminating errors and preserving schedules. Some of these are:

- Choose an object-oriented approach to reduce data coupling. If the selected implementation language does not support object orientation, use the concepts to the greatest extent possible within the limitations of the language.
- Avoid wasted effort by choosing a top-down implementation methodology. Top-down implementation minimizes what is written only to be thrown away later. A user of this methodology first creates the outer functional shell, fully populated by functionless stub modules or segments. The stubs are then fleshed out with function. There is no "scaffolding" temporary code to be discarded.

  An additional advantage is that the product itself can be the base for demonstration prototypes, providing not only "look and feel" simulation but genuine responses where they are already implemented.
- Use the proven technique of code inspection to improve the possibility of early defect removal and save future maintenance costs.
- Match top-down implementation with top-down testing of a growing functional base. Keep the testing focus on the overall project goals after modular function has been verified by unit testing.

11.7.2. Remove the stitches

- Make code segments source INCLUDEs, not CSECTs.
- Remove debugging code before putting an application in production.
- Use the most comprehensive efficiency optimizations available from compilers, including placement of code units

in-line. Take any options available to eliminate linkage code.

However, be suspicious of "optimizations" that purport to save storage. What is most likely being saved is virtual storage, a resource with no cost. Unless the compiler is new enough to provide effective optimization of real storage use, what you might get is well-folded code with a bad reference pattern, rather than code that minimizes the working set.

11.7.3. Perform volume testing in a realistic environment

- Use divergent thinkers to create test cases. This might be the perfect assignment for the lunatic hacker or the person who drives others crazy with "what-if" questions. The testing assignment should be simply to use the limits of the specifications to create cases that might break the application.
- Use performance monitors to characterize execution and delay states and to find sources of contention and execution-time hot spots during volume testing. Application programmers should be familiar with the entire set of performance management tools installed at the data center and use them to seek out areas for improvement in the finished product. Some of these tools are primarily marketed to operations groups and system programmers, although they can be of great value in understanding how an application spends its running time.

11.8. Considerations for Client-Server Applications

Client-server[4] is one of today's hot trends. It's not as hot as it was in 1993 or 1994, because the World Wide Web and Java are getting the ink in airline magazines now, but it still has some of the trappings of a fad:

- Unqualified "advocates" seek to force decisions to implement client-server applications on the basis of anecdotal evidence from stories in general-interest publications or

4 There are different ways to spell the term. "Client/Server" is the most commonly encountered variant.

trade press articles that often are little more than vendors' press releases.

- CIOs and other executives have frequently regarded client-server systems as a less expensive alternative to the presumed excessive costs of mainframes. The mainframes in those cases are invariably several generations obsolete and it might be argued that grasping the torch of client-server acts as a diversion from the executives' failure to pursue plans to keep the hardware current.
- The rise of personal computers and local area networks has led to a desire to have the LANs tap into the mainframe's mission-critical data. While *ad hoc* data serving is not true client-server, the evolution to engineered client-server applications is an obvious next step.

If the fad issues and the acts of front-runners and camp-followers can be set aside, we find that client-server is a legitimate way to structure applications. The client-server architecture simply recognizes that different platforms have different strengths. Small systems (typically personal computers) are good at providing an accessible user interface, handling data input validation, and executing guided dialogues with the user. Larger systems (typically large UNIX servers but increasingly MVS systems either in native MVS mode or OpenEdition) are tireless engines for storing and moving large amounts of data with integrity and stable performance over a wide range of demand conditions.

11.8.1. What is a client-server application?

One of the problems of pinning down client-server discussions is the difficulty of identifying just what client-server is. Consider a local area network in which some number of personal computers (referred to in the literature as clients) are connected to one or more server systems, and through less direct network connections to additional servers. The servers provide print queue services to those clients without attached printers, as well as access to files and selected applications that reside on server storage devices.

Although such a configuration has clients and servers and connections between them, it lacks one essential element of what this author regards as a client-server application: knowledge. In

an engineered client-server application, the client knows the server and communicates with it in the restricted vocabulary of a particular application; the server knows its clients and considers client actions to be an extension of the server-resident application. A frequent technique used in structuring client-server applications is to use the Open Systems Foundation's Distributed Computing Environment (DCE), in particular its Remote Procedure Call (RPC) protocol to integrate functions on the client and server platforms and to distribute them or consolidate them as appropriate.

Although client-server is good at optimizing the use of each platform in an application design (and there may be more than two levels—three-level designs are very common[5]), doing so is expensive. The RPC protocol is far more complex than a simple reference to another program in the caller's environment, and references to logic and data are executed over a network. Consequently, it is extremely unlikely that client-server performance will ever be better than performance of the same application implemented entirely on either the client platform or on (one of) the server platform(s).

The technical performance management challenge of client-server is to ensure that performance is as good as possible and that response time is consistent. The larger challenge is to contain and manage the often wildly optimistic expectations for client-server in performance, capacity, and cost.

11.8.2. Client-Server: The Customers

Client-server applications' end-users may not be experienced users of centrally supported applications. In many cases the application has taken shape on personal computers and has been lifted up to a centralized server because it has become more important to the enterprise. On the other hand, other client-server customers may have been accustomed to using a mainframe-based application with PCs as terminal emulators; now some functions of the application may have been distributed to the PCs and perhaps the platform has changed as well. In such currents of change, the users' expectations may be very confused. They may expect all the good attributes of the client plat-

5 The three (or more)-level client-server model is necessitated by the limited power of some server systems. When OS/390 is the server, such a limitation does not exist and two-level systems should become the preferred structure.

form, the power and stability of the server platform, and the operational simplicity of a single-platform application.

The management of those users may have been persuaded by "advocates" that the cost of client-server would be less than that of predecessor systems. They may have been encouraged to overlook little details like backup, recovery, security, synchronization, backup lines, adequate configuration, training.... They may also have had unrealistic expectations about the schedule to develop and deploy client-server as well as to optimize its performance.

It may not be the responsibility of the performance person to condition the users' expectations in all these areas. However, application owners and user management alike may be under severe pressure to complete the schedule. The performance person is off the hook—until the application is deployed and, more likely than not, fails to meet its performance targets. The importance of prototypes, pilot deployment, controlled scale-up, and immediate attacking of early-stage problems cannot be overstated.

If the server is OS/390, Workload Manager goal mode can help in meeting the server's portion of the performance targets. The network and the client platforms, as well as any intermediate servers, must be managed by more conventional, resource-oriented techniques. In an OS/390 environment, products such as IBM's ADSM provide a solid solution to the LAN and distributed client backup problem.

11.8.3. Client-server: the application

As suggested above, many client-server applications are adapted or converted from former applications—to "harden" an application originally spread out on uncoordinated PCs, or to "downsize" an application that has reached the end of its useful life on a back-level mainframe. The latter is often termed a "legacy" application. Typically, part of the conversion effort in the case of downsizing a legacy application is to upgrade its user interface. In the case of "upsizing" a PC-based application the more usual concern is to harden the data base subsystem.

Regardless of the origin of a converted client-server application, certain requirements are clear: changes in the business since original must be considered, new expectations and the wishlist must be respected, and an overall improvement in perceived function and performance must be seen from Day One.

For a client-server application begun as a new project, from scratch, the design job must be done exhaustively at all levels, and expectations must be managed carefully to avoid disappointment and eventual project failure.

In any case there are basic decisions to be made:

- Is it stand-alone or does it exist in a coordinated suite of applications?
- Does it rely on "middleware" or an "application framework" such as IBM's MQ Series or SAP AG's SAP R/3?
- What is the distribution of application data? Under what conditions does data migrate from a server to the next level down? What are the conditions that cause a data item to be refreshed from a guaranteed source. How much of the data is valid for a long period of time, how much is transaction-dependent, and what intermediate categories are there?
- Can the network traffic be estimated? Is there a plan to model and then prototype the data flows to get an early idea about possible bottlenecks?
- Is the deployment plan staged with enough time between stages to respond to early problems?

11.8.4. Client-server: the server

Here is where "hardware is cheap" applies—the server is often not the most expensive part of the configuration. The server platform must be robust, stable, capable of high data throughput rates, and able to handle all anticipated formats and protocols (Does this sound like MVS?)

Ensure enough capacity in the server. DASD storage and memory will be in short supply as the transaction rate builds up. CPU is usually not hit as hard.

11.8.5. Client-server: the client

If a new system is being deployed, it's important to be careful in selecting the client hardware. Usually these are ordinary PCs with modest performance requirements. (The latter is not as true if the client system is used for other purposes than a single application.) Quantity discounts and on-site service agreements

should be available. If the purchase is a large one, it may be worthwhile to buy memory and hard disks from a wholesale supplier or even from the manufacturer. Enough spares should be procured to take care of anticipated needs for urgent replacements—but not to cram a warehouse with rapidly-depreciating hardware.

Take care to match the hardware, particularly the display and video subsystem, with the most demanding frequently-used mode of the application. For instance if full-page graphics in "portrait" orientation are to be displayed, the smallest acceptable monitor for casual viewing is the 17-inch diagonal size. If 10-point text is to be read on the same display, 20-inch monitors at a screen resolution of 1280 x 1024 (horizontal by vertical) pixels or better should be specified. Just to compound the requirement, a video subsystem capable of driving the display at a vertical refresh rate of at least 76 hertz is necessary to avoid screen flicker and the eyestrain it produces.

On the other hand, 17-inch monitors are expensive and 20-inch monitors are two to three times as costly. If the specifications call for full-page views and the budget can't pay for large monitors, it may be possible to change the specifications and view

Choose the client operating environment based on the experience of most users—most OSs (Windows NT, Windows 95, OS/2, Macintosh) offer equivalent, sufficient services. Training expenses can end up higher than anticipated if the client operating system is unfamiliar to most of the users.

11.8.6. Client-server: connections

This area continues to change rapidly. In the United States, the boundaries among telephone companies, cable television suppliers, and Internet service providers have become very fluid. Formerly established territorial boundaries among different suppliers are routinely crossed. New technologies such as fiber and two-way cable offer giant leaps in bandwidth, enough to invalidate commonly-given advice about minimizing data traffic. The same forces (and others) affect the network environment elsewhere, more or less in the same way depending on the local government's degree of activism, advocacy, or intervention.

So, it is difficult to give advice in this area. If there is not sufficient skill and knowledge in communications associated with the client-server implementation project, hire the expertise.

Regardless of the kind of connections chosen, use them as efficiently as possible. Data compression is reliable and inexpensive to use. Error correction should be built in to the communication hardware.

If the communication medium includes public links, consider carefully the need for encryption. The cost is low and the potential for damage to the enterprise is great if it's not used.

11.8.7. Hypothetical Performance Example

Figure 11-1 shows hypothetical mainframe transactions terminal-based contrasted with their client-server equivalents to illustrate some points of performance concern. Effective line speeds for the client-server examples are assumed to be about one-fourth of the terminal line speeds. Example (a) is a basic CICS query and response. Example (b) is a host-driven conversational transaction, showing extra line interactions and a second host I/O block. Example (c) is the client-server counterpart to (a), showing the extra response time components. Example (d)

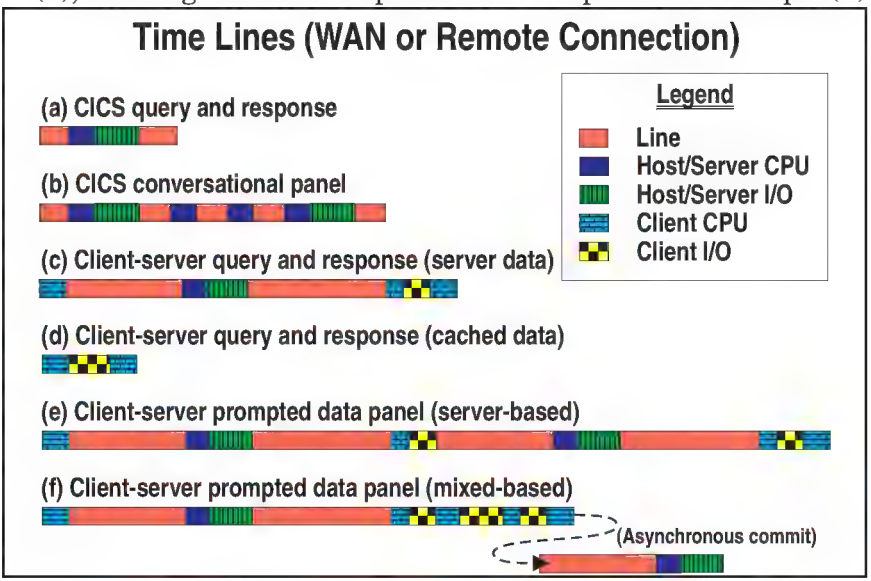


Figure 11-1. Time Lines For Various Transaction Types.

represents the rare case when a transaction can be serviced entirely on the client machine. It's really fast.

Example (e) is the counterpart to (b), where the conversation is server-based. Extra line turnarounds elongate the response time significantly. Finally, example (f) shows how response time can be cut by doing most of the prompting on the client machine. It also illustrates some possibility for overlap of transaction completion and updating on the server.

We can see from these examples that superior performance is achieved in client-server only when trips to the server can be avoided. Conversational interactions between client and server are particularly time consuming. Applications that exploit all of the hardware capability of the client platform can provide the best possible combination of function and performance.

11.8.8. Client-server: performance management

Performance management in client-server environments must consider all of the old performance factors along with a few new ones, in new and different combinations. A quick analysis makes it clear that we would deceive ourselves by monitoring just the servers; the network is an essential part of the system. If the business depends on certain client machines to be up and running, the monitoring must extend to them, but perhaps simply at the "heartbeat" level of gross availability.

Solving performance problems in client-server is complicated by the distribution of the application and by the possible multiplicity of connection types. It's therefore prudent to start as early in the process as possible, analyzing performance from the first prototype stages and adding data points as deployment proceeds to the pilot level. If the projected trend of performance appears to be unacceptable, action should be taken at the pilot stage. A well-drawn schedule will have "hold points" to allow further deployment to be delayed while early problems are overcome; the alternative is massive dissatisfaction, an abandoned project, and gross waste of resources and time.

11.8.9. Client-server: capacity planning

This topic may be an oxymoron. For an established client-server system, the usual reaction to unacceptable performance is to add a server. It's probably a less expensive action than even a minimal performance consulting contract. It's only after a few

rounds of such instinctive action without much benefit that the consultant is called in, only to discover that the network can't handle the traffic.

On the one hand, the conventional need for capacity planning—dealing with big machines that have large floor space and environmental needs, as well as long procurement lead times, is becoming less necessary. However, those old machines were run fully loaded and it was relatively easy to know which part of the configuration needed help. Client-server is different, unless the server is MVS. Then, all of the knowledge we already have of MVS systems can be brought to bear on creating appropriate capacity plans for the servers. The undemanding needs of clients will usually be less of a problem.

Other aspects of client-server, including training costs, disaster recovery, backup, currency, and synchronization concerns, are beyond the scope of the performance management theme of this book. They must be addressed, but this should be done in the context of a more comprehensive view of enterprise computing operations and economics.

11.9. Practical Approaches to Cooperation

The organizational tensions that often exist between the data center and the applications staff are not eased when a performance analyst makes a vague "suggestion" that a particular application program is the cause of poor response time, as opposed to something in the way the system is managed. The more detailed the analysis, the more specific the recommendation can be, and the more likely that it will be accepted. Results of execution-state analysis and direct contention analysis can be easily understood by application developers or subsystem maintainers, and corrective responses are clearly suggested by the kind of information presented.

It can be difficult for someone without specific responsibility for an area to make credible recommendations for change in that area. A cool, factual, limited, and respectful approach is much more likely to be successful than the kind of emotional and confrontational scene often engendered by a service crisis. This advice may sound obvious and superficial, but ego involvement can often drive people out of control, particularly when solid facts are not available when required.

Therefore, get the facts! When a manager rages that Accounts Receivable came up late for the last two days, the last thing needed is confirmation that the problem did indeed occur. What is required to defuse the situation is a clear explanation of why the batch update was late (what caused it?) and a credible plan to avoid it in the future. (Credibility is compromised already because the problem should have been identified and headed off the day before.)

Workload-oriented data collection tools, preferably with contention analysis capabilities, are essential for this task. Real-time alerts based on overdue production batch could have made remedial action possible within whatever buffer might have been available. It takes far more factual data, credibility, and presentation skill to calm down an irate manager than to put a plan and procedure in place to deal with late batch jobs.

11.10. Summing Up

How much is it worth to practice successful performance management in MVS? Hardware acquisition can be avoided or deferred; performance can be made acceptable and consistent. The new challenges of client-server and network-centric computing can be dealt with in a straightforward way, building on the strengths of a polished methodology. Perhaps most important, though, the human resources formerly devoted to creating useless reports, making accusations, and finding fault can be directed to productive tasks.

11.11. Chapter Questions

1. Identify the "top 10" applications in your installation. What was the basis of your selection? Go back and make sure it's on the basis of importance to the business or enterprise.
2. Now name your five most troublesome applications. What kind of trouble do they cause? Are they all on the "top 10" list? Rework the list to identify the worst five of the top 10. Now eliminate any whose users (customers) are completely satisfied with current and projected performance and cost. Redo the list to include the worst performers of the top 10 which are perceived as problem areas today.

   If you can't come up with any examples meeting all criteria, look for any parts of your workload about which there

are complaints. If you still can't find any, you may have that rarest of systems with no performance problems. Make some up by projecting credible significant growth in volume or complexity.

3. Develop an ideal action plan to overcome the problems of the applications you listed in response to question 2. Identify who is responsible for each item. An ideal plan costs no real money and shows self-evident value at each stage to each participant.
4. If ideal (no-cost) plans are not possible, identify the additional resources with the minimum cost necessary to deal with all five problem areas. How did you arrive at your estimates? What measurements and analysis make you sure? What else would you prefer to have?
5. Now sell your plan, organizing it as a set of tasks for each decision maker and affected individual or department. Selling it can be easier if you stress your involvement and willingness to work with the responsible individuals to find optimal solutions.

Chapter

12

Summing Up and Looking Ahead

12.1. The MVS Journey—How Much Farther to Go?

When OS/360 was announced in 1964, no one could have foreseen what was to come. One megabyte of processor storage was seen as "huge," processor cycle times were measured in microseconds, and a "large" storage device capacity was less than 50 megabytes. Today, run-of-the-mill personal computers exceed those figures by orders of magnitude.

Yet System/360 and OS/360 changed the world of computing and redefined the term "operating system." Although other substantial operating systems have been implemented since 1964, they have been influenced one way or another by OS/360 and its progeny. In some cases, the influence is direct. OS/2, for instance, shares many concepts with MVS.

In other cases, the influence is one of aversion. The Macintosh operating system attempts to be sophisticated and simple at the same time. Virtually all the terminology of MacOS seems to be invented to differ from that of MVS. Of course, the archetype of the anti-IBM operating system is UNIX.

In the twenty-first century, operating system games must end. The set of common services expected of any operating system must be offered by a descendant of today's MVS, in a way that is compatible with other offerings, both from IBM and its competitors. The underlying internal structures will provide the differ-

entiation across different implementations. OpenEdition MVS is a good start in this direction.

Good ideas that have evolved in other arenas, such as plug-and-play and graphical user interfaces, should be seen in the enterprise computing world as well. To the extent that the computing model of the future is network-centric, the network must be made invisible—*i.e.,* such a well-integrated part of the computing environment that one is not aware of network connections or cryptically coded IP addresses. Instead, the user attempting to connect to a remote server simply would need a name, like today's ubiquitous "www.*anything*.com."

The power and capacity of today's hardware and the expectation of continued improvement in price/performance and raw power makes it imperative to revise some of the underlying beliefs of data processing. It makes no sense whatever to require human beings to perform tasks that can be done equally well or better by programs. It is indefensible for a vendor to provide a hardware device that is not self-revealing and self-installing when connected to a system with a well-defined plug-and-play interface.

It follows that an operating system and the hardware it runs on must facilitate and support automatic detection and installation of an attached and ready device that is not already known to the system. (The exception is to permit devices to be logically disconnected when service actions are required.) Continuous operation is a given; nondisruptive servicing is a given.

None of these initiatives require invention but simply adaptation of what is already commonplace in contemporary systems. The most pressing need is for standardization of the attachment interface at a high enough level that it might last for 30 years or more.

12.2. The Ideal Large System

Some time around the year 2010, we might expect to see enterprise computing running on hardware that is as much smaller than today's MVS CMOS machines as those machines compared to their ECL bipolar predecessors. The new hardware would be 20–100 times faster and 100 times larger in memory capacity than today's systems—and it would cost less than one-tenth as much in today's currency. Online storage would be a nonmoving technology such as holographic images with capacity of a few

terabytes and an access time of a millisecond, and backing devices in the petabyte or even exabyte range.

The hub server would be connected to its users by fiber-optic or wireless connections with effective data rates of 100–1000 megabytes per second. The operating system would provide a graphical user interface which could be enriched or replaced by a different interface at a work station. Information would travel between server and clients in object-oriented, device- and interface-independent form, possibly derived from today's HTML and Sun Microsystems' Java.

The code resident on each client work station would be propagated from the hub server at times of inactivity. Similarly, distribution of maintenance from the vendor to the customer would occur in the same manner. Since the transmission is such cases is primarily one-way, media such as satellite broadcast might be useful in this area.

As capacity and bandwidth increase, the need for intermediate servers would diminish; simple two-level models would suffice except when span of control is a significant manageability issue.

The user interface of the work station would be presented on a thin panel device (possibly flexible and portable) with resolution exceeding 1600 by 1280 pixels at whatever screen size is convenient. Keyboards will still exist, but will be supplemented by voice communication for most command and noncritical data entry functions. Local data storage in the work station might be needed to support alternate GUIs, and if off-network operation is likely, but bandwidth and capacity in the server and network diminish the value of client data caching.

Continued evolution of the user interface and adoption of paradigms like bookmarks and cutting out clippings will cause paper output of nonarchival documents to diminish greatly. Legal changes may make further reduction of paper output possible, as technology makes it possible to create unalterable permanent records in electronically viewable form, with alternatives to current facsimile (fax) systems capable of high-resolution color transmission. Text management including font management would be integrated deep in the operating system, instead of being relegated to a printing subsystem.

The user's screen device would show full-motion video in lifelike color; holographic projection devices will render three-dimen-

sional images. Application programming would evolve from today's object-oriented and visual methods, with the addition of transparent global optimization capabilities to ensure that the gains in processor power are not squandered by inefficient programs.

12.3. How Does MVS Meet the Ideal?

MVS, as embodied in the early releases of OS/390, is moving in the right direction but still has a long way to go. The re-integration of much of the operating system is a good start; next should come the rationalization and integration of subsystems such as DB2 , CICS, and JES.

Adding hardware interfaces for painless reconfiguration should be within today's capabilities; revising the instruction set and structure to exploit terabyte memory sizes might be more challenging. The existing MVS real memory structure is not the problem. Real memory can be hidden. Revealing a new structure for MVS virtual memory will be an appropriate step by 2001, the twentieth anniversary of the announcement of MVS/XA. An evolution from today's parallel sysplex might provide a path to a higher-level cellular architecture.

The next challenge for MVS will be the integration of graphical user interfaces into the operating system. In this case, there is little technical challenge but a great paradigm shift is needed.

An early bad decision by IBM was that of clinging to proprietary EBCDIC instead of ASCII as the internal character set. As UNICODE becomes an accepted standard, IBM has another chance to join the world and remove an objection to a choice of IBM as the system vendor.

12.4. The Ultimate Wish List

I wish IBM would continue the evolution of MVS in the positive direction begun in OS/390.

I wish that the intrinsic capacity, performance, and price-performance of the future MVS platform will make performance management concerns simply those of ensuring that adequate capacity is available. Of course, that means that the self-management and self-optimization of MVS performance will grow well beyond what is in today's Workload Manager.

I wish that the functional capabilities of MVS and its subsystems will be so well integrated that users in the future will think only of interacting with their data, not with the operating system or its obscure subsystems.

Now, Dear Reader, what is your wish?

Appendix

A

SRM in Compatibility Mode—Concepts

A.1. Application to Workloads

The concepts developed in Chapter 6 are of little use by themselves. In this appendix we examine the interaction of those ideas with the work that runs (as address spaces) in an MVS system. In compatibility mode,[1] address spaces are organized by the SRM in two ways:

- A *performance group* defines many operational parameters associated with each address space, including its dispatching priority and storage isolation. The performance group definition also identifies which domain will be in control of each of its address spaces at any time.

- A *domain* defines a group of address spaces to be managed for swapping according to specific rules as directed by SRM, to control system loading and apportion resources. The domain definition may also denote a set of criteria (de-

1 In Appendixes A and B, "compatibility mode" refers specifically to MVS Version 5 and OS/390 compatibility mode. Advice in these appendixes is broadly applicable to Version 4 (SP 4.2 and later), but certain parameters have been eliminated or redefined in compatibility mode. A reader desiring a more complete treatment of the Version 4 SRM parameters is urged to obtain the prior edition (*ESA/390 Edition*) of this work.

fined in the OPT) that determine expanded storage eligibility for the pages of each address space in the domain.

A.2. Performance Groups

Each address space in MVS is assigned to a performance group, either explicitly or implicitly. *Control* performance groups (those defined in the IPS) exist for the purpose of allowing SRM to manage the attributes of address spaces. These include:

- domain
- dispatching priority and I/O priority
- storage isolation
- performance period control

Each of these attributes is examined in detail in this appendix. The last attribute might remind us that each performance group is defined as one or more (up to eight) performance periods, so that treatment may change as transactions age. A nonswappable address space assigned to a multiperiod performance group always stays in the first period. Documentation for MVS/ESA SP Version 5 (*Conversion Notebook*) states that nonswappables may pass through multiple performance periods, but the OS/390 *Initialization and Tuning Reference* does not say so. (If an address space becomes nonswappable during execution, it stays in whatever period it was in at the time of transition.) For swappable address spaces, service or time accumulation determines the passage from period to period; a new transaction always starts in the first period.

A.3. Service Intervals

SRM bases some of its actions on the amount of service an address space receives during some period of residency. All of these decisions are related to swapping, so the values discussed below apply only to swappable address spaces.

A.3.1. ENQueue residence value (ERV)

ERV is a means of ensuring that an address space holding an ENQueue on a resource for which there is current contention is protected from being swapped out during a specified interval.

ERV is measured in CPU (TCB only) service units, has a default value of 500, and is specified in the OPT. This parameter survives from the earliest days of MVS. It has been updated as of the introduction of enclaves to include those work units, defining the interval during which they execute at the privileged dispatching priority (PVLDP). Address spaces within their ERV service durations execute with PVLDP as well.

A.3.2. Performance periods

SRM can manage swappable transactions differently at each stage of aging. The measure of that aging is usually accumulated service, and the means of changing management parameters is the passage from one performance period to the next. The DUR parameter gives the duration of each performance period except the last in a performance group, and is usually measured in total service units (SUs). A rarely specified UNT=R parameter in the performance period definition allows DUR to be measured in wall-clock seconds instead of service units. UNT=S (for Service) is the default and need not be specified.

TSO performance period DURs usually range from a few hundred SUs in the first period to several thousand in the next-to-last (third or fourth) period. (The last period is always unbounded, and DUR may not be specified.) A typical batch first period might be 5000 service units; the second period might be 20,000 to 50,000, if three periods are used.

The use of performance periods sometimes becomes a perplexing part of MVS performance management. If periods are used only to recognize different types of batch and TSO transactions, there is little complexity. A simple clustering analysis of completion statistics for a representative sample of transactions might show a few natural clusters of transaction sizes. If 81 percent of TSO transactions have 200 service units or fewer, and very few end in 200 to 400 service units, it seems reasonable to set the duration of first period to some value between 200 and 400. The same rationale can be used to establish groupings of batch job durations and set up two or three batch performance periods for unscheduled batch. (Production batch may not require the same treatment; multiperiod performance groups may not facilitate consistent turnaround times and instead serve to stretch out longer jobs.)

When performance periods are used for other purposes than supporting the natural divisions of transactions, the subject be-

comes complex and works against efficient operation. It may be very comforting to report that 85 percent of TSO transactions are "trivial" and therefore complete in first period. It is plain silly to require this to be so. The work will be what it will be. The purpose of performance management should be to handle the received workload efficiently and effectively subject to the needs of the business, not to warp it into something else. Of course, if "natural" efficiency and effectiveness are consistent with business reality, so much the better.

A.4. Domains

Each address space is assigned to one domain at any time. If an explicit assignment is omitted in the performance period definition, a default assignment of domain 1 is made for ordinary address spaces and domain 0 for privileged address spaces. Privileged address spaces include initiators, LOGONs, VTAM's Terminal Control Address Space (TCAS), and those with an active program named in the *Program Properties Table* (PPT) with the PRIVILEGED attribute.

Each domain (other than domain 0) is defined in the IPS. Although only the domain number must be specified, two additional kinds of information are usually supplied: constraints and a means of setting the contention index. An additional parameter designating a set of expanded storage eligibility criteria may be added as well.

A.4.1. Constraints

Domain constraints define the range of allowable MPLs for a domain and are specified as a lower limit (minMPL) and an upper limit (maxMPL) in the range of 0 through 999. The limits may be equal. The current MPL of a domain includes both swappable and nonswappable address spaces.

When the upper and lower domain constraints are equal, control of the domain's MPL has been denied to SRM. Constraints of (0,999), on the other hand, allow full SRM control. The default, however, is (1,999). A nonzero minMPL may be specified to ensure minimum swap-in delay for TSO users or swappable subsystem address spaces. A restrictive maxMPL allows the installation to limit the impact of resource-intensive or contention-prone jobs.

Prior to MVS/ESA SP 4.2, there was little benefit in having more than about ten domains.[2] Swap management based on intra-domain contention became ineffective when work was split in domains matching performance groups, and SRM overhead was very slightly increased.

As of SP 4.2, there are new, much better, reasons for having additional domains. Because there are no longer OBJs in performance period definitions, there is no equivalent to associating multiple performance periods on different OBJs with a single domain. Each such workload segment now needs a different domain. Additional domains are also needed to specify expanded storage criteria for individual nonswappable address spaces, and to differentiate among different swappable workloads in their eligibility for access to expanded storage.

TSO is usually an important swappable workload. Typically, there is a separate domain for each of the first one or two performance periods, and another covering the last period or two. A typical specification for a domain associated with a TSO first period is to set the minMPL equal to about one-fifth of the number of logged-on users expected at an average peak period. The first-period maxMPL might be set to the approximate number of TSO address spaces that could be accommodated in real storage, assuming that batch MPLs are driven to their minimum values. (If more specific information about TSO working set is not available, a working set of 100 page frames per TSO address space might be assumed for this calculation.) Such a maximum might be needed to deal with the aftermath of a TSO stall condition such as an ENQ lockout of a commonly used shared data set.

An APPC/MVS workload serving many users in individual transaction processor (TP) address spaces has similar characteristics to TSO. Its domains should also have minMPLs adequate to minimize avoidable swap-in delay. A similar treatment applies to "batch" domains serving an interactive workload like CADAM.

Other domain MPLs should be defined to be as close to their default wide-open values as possible. In other words, trust SRM! If SRM is allowed to adjust the system-wide MPL based on appropriate criteria, distinctions among domains can be made by se-

2 Before report performance groups became available in 1978, there was some justification for segregating workloads in domains for (RMF) reporting purposes.

lecting differing service-rate ranges for setting each domain's contention index. It may be difficult to get it right, and it may be somewhat intimidating to yield control to SRM if one has not done so before, but the alternative is to remain continually involved in trimming and adjusting inflexible domain constraints in order to keep current with workload changes. Choosing this course is not a good career decision.

A.4.2. Contention indexes since SP 4.2

In SP 4.2, the ASRV and DSRV parameters became the only means of setting the domain's contention index. [3] In each case a pair of service rates establishes the points corresponding to contention indices of 1 (at the high service rate) and 100 (at the low service rate). A simple linear interpolation of the current service rate gives the current contention index. Out-of-range service rates give contention indices less than 1 or greater than 100, to a maximum of 655.35 at zero service rate and a minimum of 0 at infinite service rate.

To make a domain's MPL dependent on its total received service, use DSRV. There should be a substantial difference between the low and high values. The lower the high value, the lower the contention index for a given in-range service rate will be, and the more likely the MPL will go to its minimum. The higher the low value, the higher the contention index for a given in-range service rate will be, and the more probable it will be for the MPL to go to its maximum. ASRV works the same way, except that the *average* service rate (DSRV divided by the Ready User Average, RUA) is used in the determination.

Here are some examples of possible domain MPL control parameters for various workloads:

- First-period TSO for 300 logged-on users:

```
DMN=4,CNSTR=(60,120),FIXCIDX=115
```

  Up to 60 ready users should experience no MPL delay as they enter new transactions. Additional users, up to the maximum of 120, will be swapped in as they become ready. Such a large number is unlikely except after relief of a stall condition.

3 In a subsequent release, a FIXCIDX parameter was added to once again provide for a fixed contention index, previously specified as FWKL, a parameter dropped in SP 4.2.

- Second-period TSO and first-period batch:

  ```
  DMN=9,CNSTR=(9,50),ASRV=(2000,10000)
  ```

  The average service rate should fall between the ASRV limits, allowing MPL adjustment. The minMPL of 9 should allow transition without swap of the approximately 15 percent of transactions that go past first period, and allow 2–3 batch jobs to start.

- Third-period TSO and second-period batch:

  ```
  DMN=33,CNSTR=(0,20),DSRV=(3000,30000)
  ```

  This domain has a potentially high impact, even from a single address space. The MPL is allowed to go to zero when system loading forces an MPL reduction. Specifiying 30,000 as an upper service limit ensures that this domain will have a low contention index when a few address spaces are resident. On the other hand, when there is no contention, as many as 20 jobs or long-running TSO transactions are allowed in central storage.

- Nonswappable domain:

  ```
  DMN=15
  ```

- Swapout domain:

  ```
  DMN=13,CNSTR=(0,0)
  ```

- Low-priority batch:

  ```
  DMN=6,CNSTR=(0,40),DSRV=(0,20K)
  ```

- High-priority batch:

  ```
  DMN=10,CNSTR=(0,30),DSRV=200K,500K)
  ```

Note in the last two cases that the high-priority domain has a lower constraint range than the low-priority domain. The DSRV range is what controls contention index, not the constraints.

A.5. MPL Adjustment

The principal reason for swapping out an address space in MVS is to ensure that sufficient central (real) storage is available to service page faults and swap-ins. Swapping may be regarded as the *macro* control of the available frame queue. Page stealing is the *micro* control. Since swapping can be less disruptive and more efficient than page stealing, a goal of performance management should be to minimize the incidence of page stealing.

Swapping affects only one address space at a time; page stealing may cause deferred damage to many address spaces as page faults later occur. However, stealing very old pages may be the least disruptive choice at many times. The choice of which mechanism to encourage is complex at best.

IBM recognized that independent control of page stealing and swapping can lead to uneven performance. In MVS/ESA, and especially since SP 4.2, the control of paging and swapping is highly integrated, and further integrated with the operation of the Working Set Manager (WSM). In general, the only reason for pages to be moved out of central storage is because the system can operate better with them otherwise assigned. (There are a few rare exceptions such as swap-outs to handle or prepare for various kinds of reconfiguration.)

All stolen pages, swap trims, and swap working sets stay in central storage until the frames are needed. When that need exists (as determined by the exhaustion of the Available Frame Queue), the SRM determines how to meet the need. If the average system page age (UIC) is relatively high (above about 200 seconds), page stealing is the mechanism of choice. Address spaces selected for management by WSM come first in selection for stealing, along with those storage-isolated address spaces holding frames above the maximum protected threshold. Ordinary page stealing then proceeds if frames are still needed.

In systems with lower UICs, a generic activity called pushout supplies the needed frames. Pushout is the process of moving blocks of pages from central storage to the next lower level in the hierarchy, either expanded storage or auxiliary. Pages having both equal UICs and adjacent virtual addresses move as blocks.[4] When a swappable address space is left after this process with only its primary working set (or its swap group if it was to be a direct swap to auxiliary storage), a swap-out takes place the next time that the same address space is asked to donate pages to the needy.

In a system with expanded storage, there is little need to choose between swapping and page stealing unless the amount of expanded storage is inadequate to meet the needs of all workloads. With such a constraint, selected pages of particular workloads

4 Explicit block paging is available to application programmers, and allows them to define nonadjacent address patterns for block paging.

may be excluded from expanded storage, either absolutely or contingent on a shortage of expanded storage frames.[5] The excluded workloads are then managed (for MPL control) exactly as if expanded storage was not present.

If the system has a significant swappable interactive workload (such as TSO), most swap-outs are of inactive address spaces. Physical swap-outs will eventually occur (in systems without expanded storage) when the criteria for logical swap are not met. With expanded storage, physical swap-outs go to expanded storage as central storage frames are needed, or to auxiliary storage (again only when the frames are needed and after trimming has reduced the swap group to a manageable size) when the relation between migration age and workload-related criteria (specified in the OPT and selected by domain) indicates that expanded storage should be bypassed.

SRM initiates swap-outs or swap-ins of active address spaces when its system measurements indicate that at least one controlled variable is out of the desired range.[6] By computing several measures of system activity each SRM second (RM1) and summarizing them at a longer interval (RM2), SRM decides how to adjust the system-wide multiprogramming level (MPL) and recalculates the target MPLs of each domain. The single target MPL of prior MVS releases was replaced in SP 4.2 by two targets. The *in-target* (ITRG) MPL is the minimum number of address spaces to be resident under conditions of resource constraint. The *out-target* (OTRG) MPL is the maximum number of address spaces to be allowed in central storage when there is not a resource constraint. The various MPL numbers for a domain in SP 4.2 or later MVS systems follow the sequence:

low constraint (minMPL)
≤
in-target
≤
current MPL
≤
out-target

5 Use the ESCRTABX parameter on the DMN statement in the IPS to select the appropriate set of expanded storage criteria in the OPT. The details are in Appendix B.

6 A simplistic view is that the resources are "over-" or "underutilized."

$\leq$
high constraint (maxMPL)

SRM recalculates in- and out-target MPLs for the domains, based on current domain MPLs and contention indices. The set of targets is one of the inputs to the Swap Analysis process that is at the heart of SRM-directed swapping.

Swap Analysis also regulates swap-ins, both those initiated by SRM's resource management actions and those needed to reactivate previously inactive address spaces. Active address spaces swapped out by SRM and newly ready address spaces, swapped out when inactive, all compete for admission to the multiprogramming set. Each competes only in its own domain, so the more important initial Swap Analysis decisions deal with domains.

A.5.1. The Swap Analysis cycle

Swap[7] Analysis is initiated each SRM second, at the completion of each swap-out, and whenever a swapped-out TSO address space becomes ready[8] (*i.e.,* the user hits ENTER). At each activation, the following actions take place, subject to the usual caution that IBM does not guarantee not to change MVS (and Swap Analysis in particular) at each release or even with each maintenance update:

- Swap Analysis retries swap-ins previously deferred for lack of frames.
- It then swaps out address spaces in each domain currently exceeding its out-target. Enough address spaces are chosen (those with the lowest RVs) to make the current MPL in the domain equal to the out-target. The unilaterally swapped address spaces are retained in central storage until their frames are needed. (Pushout is deferred.)
- If an address space holding an ENQueue is swapped out, Swap Analysis swaps it in. If its domain's current MPL is

7 Don't confuse the swapping process with the need (or lack of need) for swap data sets. As suggested in the discussion of MVS evolution and history in Chapter 1, swap data sets are obsolete and should not be used except as a last resort or in very unusual circumstances. An extended discussion may be found in Chapter 10.

8 The SRM reacts to the USRREADY SYSEVENT issued by the TSO Terminal Monitor Program.

at its out-target, the domain's swapped-in address space with the lowest RV is swapped out to maintain the domain's MPL. The swap-in is an ENQueue swap-in, with or without an *ENQueue exchange swap-out*.

- Swap Analysis swaps in address spaces in each domain having at least one out-and-ready address space, and currently at an MPL less than its in-target, up to the number needed to restore the in-target for each domain.
- If there is remaining out-and-ready work, at least one domain's MPL is below its out-target, and if there are remaining central storage frames, additional swap-ins are performed, beginning with the domain having the highest contention index and swapping in one address space per domain until the available and discretionary frames are gone.[9] A swap-in in this case may simply mean moving an address space to the dispatching queue from the queue of "logical out-and-ready" address spaces in processor storage swap; in such a case no frames are needed.
- As its last priority, for each domain (in the same order) with both an out-and-ready address space with an RV greater than 100 and a swapped-in address space with an RV less than 100, Swap Analysis performs an *exchange swap*, interchanging the highest-ranking "out" and lowest-ranking "in" address spaces.

A.5.2. Implications of Swap Analysis

The foregoing synopsis of Swap Analysis suggests some performance management implications:

- The primary attention given to swap-in fails may supply an explanation for swap-in delay, even in domains with generous minMPLs. If this condition is seen frequently, SRM's MPL controls in the OPT could be adjusted to reduce the system-wide MPL, thus increasing the supply of available frames. This problem was nearly eliminated with the introduction of Working Set Manager in SP 4.2. As with several other SRM algorithms, SRM must not be denied the discretion to reduce MPL, so most minMPLs should be kept low—zero or one.

9 Discretionary frames are those eligible for immediate push-out.

- Unilateral swap-outs inhibit unilateral swap-ins in other domains until at least the next Swap Analysis cycle. If MPL targets change too frequently (causing a disruptively high level of unilateral swapping), perhaps overly sensitive controls, such as too narrow a range of service rates in an ASRV or DSRV, are to blame. Also, breaking a workload into several similar domains with small MPLs may lead to more frequent unilateral swap-outs. Consolidating domains with similar workloads could prove more stable.

A.5.3. Types of swapping

In discussing Swap Analysis, we touched on several types of swaps. We'll review those now and list other types as well.

A.5.4. Unilateral swapping

This is the type of swapping initiated at frequent intervals by SRM so as to move the MPL of each domain toward its current in- or out-target MPL. By controlling each domain's MPL, the overall MPL is controlled, and in turn the measurable factors (such as page fault rate) used to control MPL are themselves controlled. In this way, SRM's MPL adjustment actions constitute a classical negative feedback control system, something like a thermostat controlling a heating or cooling system based on a temperature measurement, in turn controlling the temperature in the monitored space. When unilateral swapping of address spaces that are not performance-critical occurs, and when that swapping is controlled by the page fault rate, less page stealing and consequent page faulting is likely to occur in other workloads.

Response-oriented workloads such as short TSO transactions should not normally be unilaterally swapped out. On the other hand, a batch job may be unilaterally swapped many times before completion. As long as batch turnaround-time targets are safely met, unilateral batch swapping is a sign of sound installation control of SRM's workload management.

The absence of such swapping (in a system with a sizable swappable workload and some degree of storage constraint) may indicate a system likely to incur a great deal of demand paging delay. Such behavior may be caused by the presence of one or more domains with overly high minMPLs. Again, demand paging (page stealing followed by page faulting) is the less efficient

way to maintain the available frame queue when system MPL control through batch swapping is an available alternative.

A.5.5. Exchange swapping

The purpose of exchange swapping is to allow several batch jobs in a chronically "overloaded" domain to have nearly equal service rates on average over a reasonable time frame. Without exchange swapping, the elapsed time of a short-running batch job may be at the mercy of a longer-running job in the same domain that happened to have been initiated first.

An exchange swap will occur on an otherwise unproductive Swap Analysis cycle *only* if the following conditions are met:

- A domain is exactly at its (nonzero) out-target MPL.
- At least one address space in the domain is "out-and-ready."
- At least one resident address space in the domain has a swap RV of less than 100, indicating that it has received enough service to justify the cost of a swap-out.
- The swap RV of an "out-and-ready" address space is greater than 100, meaning that it has been out long enough to justify the cost of a swap-in.

Such a series of conditions is rarely met, yet exchange swapping was so feared in the early days of MVS that extraordinary measures were taken to prevent it.

The factors making exchange swaps undesirable in the past no longer exist, and the original rationale for allowing them still holds. In installations with significant batch or long-running TSO workloads, encouraging exchange swapping can result in more uniform completion times for such transactions.

Often stringent service targets exist for unscheduled batch, yet resource constraints dictate a restricted MPL for all or part of the batch workload. The use of a low SWAPRSF (less than 10) can, by encouraging exchange swaps, ensure a consistent degree of elongation and less variability in turnaround time for short-running jobs, regardless of the arrival order of large and small jobs in the multiprogramming set.

A.5.6. Swapping of inactive address spaces

As we saw in Chapter 1, much of the evolution of MVS involved changes in the swapping of inactive address spaces, particularly TSO address spaces. Let's look at the progress and decision points of such a swap in today's MVS:

- The *inactive* condition is detected by SRM. The TERMINAL WAIT SYSEVENT (usually for input, sometimes for output) may have been issued by the TSO Terminal Monitor Program (TMP), a program may have issued the *WAIT* macro-instruction with the *LONG* operand, or an ordinary *WAIT* may have continued past the *detected wait* threshold. The threshold is 2 seconds or 8 SRM seconds, whichever is greater. The former threshold is effective in all but the slowest systems. The 2-second value may be altered as described in Appendix B.

The distinction between terminal waits and other types of inactive address spaces will be important in the subsequent decision process.

- The address space is prepared for swap-out. First, it is *quiesced*, ensuring that all pending I/O is complete. Next, pages not in the working set are *trimmed.* (Swap trim is a special form of page stealing from the address space about to be swapped out.) A few trimmed pages may be *unchanged* if they are backed by valid copies on auxiliary storage. The unchanged trimmed pages are added directly to the available frame queue (AFQ) in systems without expanded storage.

  Trim may take place in stages to hold down the system impact of a large address space being swapped out. An OPT parameter, MCCMAXSW, establishes the maximum number of pages to be moved out at once. Its default value is 512 pages, or 2 megabytes.

  The *working set* includes pages with the hardware reference bit on, fixed pages, and the local system queue area (LSQA) of the address space. This "strict" working set is "enriched" for TSO address spaces by the addition of pages with UIC of zero, and of additional pages as the system-wide high UIC increases.

For systems with expanded storage, the working set is broken into primary and secondary subsets. The primary working set includes LSQA, fixed pages, and one page from each segment in the address space. The secondary working set is all other working-set pages. Pages not in the working set will be trimmed in theory but not necessarily moved out of central storage unless their frames are needed and no more likely source of those frames is available.

With expanded storage, trimmed pages are evaluated against a criterion for movement to expanded storage. The few unchanged pages are released to the available frame queue. The *changed* trimmed pages normally will be placed on a queue for movement to expanded storage, grouped in blocks according to UIC and virtual address. They are moved out if and when central storage frames are needed to replenish the available frame queue. Without expanded storage, the pages are scheduled for [blocked] page-out to auxiliary storage. Again, these frames are added to the AFQ once they are successfully backed in expanded or auxiliary storage.

- The eligibility for logical swap is evaluated next. For TSO terminal waits, if the recent think time of the address space is less than the current system think time threshold,[10] the address space is eligible for a logical swap. Other inactive address spaces are eligible for logical swap if the current system think time is 5 seconds or more.
- If the address space is eligible for logical swap, the logical swap decision is then made, according to the system's ability to accept a logical swap at this time. If the system-wide high UIC is not less than one-half the UIC lower limit for think time adjustment (LSCTUCTL), the logical swap will proceed. The address space is then dropped from the dispatcher's true-ready queue, and other control block adjustments are made to indicate that the address space is "OUT–LOGICAL."
- If logical swap is not selected, and expanded storage is lacking, or if the workload is excluded from expanded storage, a physical swap to auxiliary storage begins.

10 The logical swap think time is determined by SRM measurements against three value pairs specified as OPT parameters. They are described in Appendix B.

- With expanded storage, address spaces that will not be logically swapped are evaluated for direct swap to expanded storage. The OPT parameter ESCTSWWS(2) specifies a default *criteria age* for the working sets of TSO address spaces in terminal WAITs. ESCTSWWS(0) sets the default for swaps of privileged address spaces, and ESCTSWWS(1) determines the default criterion for all other swaps.[11] For other than terminal wait swaps, the swap to expanded storage (known generically along with logical swaps in central storage as a *processor storage swap*) will be selected if the sum of the current expanded storage migration age (MA) and the system-wide high UIC exceeds the criterion. For terminal wait swaps, the address space's think time is added to the criteria age before the comparison is made.

  The working-set pages of an address space selected for a processor storage swap to expanded storage are placed on the FIFO queue of pages (we'll call this the *pending-out queue*) to be moved out when real storage frames are needed. The entire trimmed working set will be moved out together, after all trimmed pages have been moved in UIC-grouped stages as pages are needed.

- If expanded storage is lacking, or if the criteria-age test for expanded storage swap eligibility fails, the trimmed working set is moved together to swap data sets if they are present, or to local page data sets, using the contiguous-slot allocation algorithm, if possible. Again, this movement occurs only when the frames are the next candidates to replenish the available frame queue.

At this point the address space is either logically swapped in central storage, physically swapped on auxiliary storage, moved out to expanded storage, or awaiting such movement. Let us assume that the processor storage swap to expanded storage proceeds. We can now drop from consideration the direct swap to auxiliary storage. Nothing more happens in that case until the address space becomes ready for swap-in.

- A logical swap in central storage stays there until the expiration of the system think time. (In MVS releases prior to

11 This OPT parameter and related ones can have index values ranging from 0 to 99, with the ESCRTABX(*n*) parameter on the DMN statement in the IPS designating one of the alternate criteria sets.

SP 2.1.7, an arbitrary 15-second "grace period" was added on to this interval. The grace period in current releases is 2 seconds.) At this time the logical swap fails, and the address space now receives the same treatment that it would have had if it had failed the test for logical swap at the outset.

- An address space swapped to expanded storage stays there until it is swapped back in, unless migration reaches it. Migration takes place in stages, with the secondary working set being moved first, in groups of pages according to the device type of the local page data sets. Finally, the primary working set is moved to the swap data set(s), if present. If there is no swap data set or no available swap set, the primary working set goes to local page data sets.

  The impact on system and workload performance of expanded storage migration can be considerable; each migrated block of pages requires central storage to accommodate the move-in from expanded storage. The central storage frames are kept for the duration of the subsequent physical I/O to auxiliary storage as well. Since migration (a symptom of expanded storage constraint) may occur when central storage also is constrained, this added storage demand with migration can cause a sharp degradation in the performance of workloads sensitive to paging delay. A temporary cure for disruptive migration is to increase the values of the OPT parameters (those named ESCTxxxx) that control expanded storage eligibility; the long-term cure is to increase installed expanded storage. In systems capable of logical partitioning, this increase may be accomplished by reapportioning expanded storage across partitions.

  Another option became available with MVS/ESA SP 4.2: *selective* increase of the expanded storage selection criteria, thus restricting the use of expanded storage to the most important workloads. Details are presented in Appendix B.

- Some time later, the address space becomes ready. (In the usual case of TSO, the terminal user has pressed ENTER.) A logically swapped address space still in central storage is ac-

tivated by queue manipulation, which reverses the actions taken at the time of swap-out. A similar reinstatement takes place for an address space on the pushout queue.

- An address space that has been physically swapped out may be in expanded storage, in auxiliary storage, or in transition between the two owing to migration. In each case, the swap-in is *unilateral* according to the Swap Analysis procedure described previously. Until Swap Analysis permits the swap-in, the address space is counted as out-and-ready. Migration from expanded storage to auxiliary storage (through central storage) may still take place during the out-and-ready period.

The progress of a swap is complex. Obviously, the best case for performance is the successful logical swap in central storage, or the unfulfilled processor storage swap to expanded storage. When response-critical address spaces are physically swapped to auxiliary storage or (worse) migrated from expanded to auxiliary storage, they are subject to both I/O delays and possible delay for SRM MPL adjustment. Much tuning activity can be devoted to overcoming both kinds of swap delay.

A.5.7. Miscellaneous swaps

Several other types of swaps can occur in MVS. Normally, they are all very infrequent and have no effect on performance. Two types triggered by storage shortages, however, may indicate serious resource imbalance problems. The types of miscellaneous swaps are:

- **Request Swap**—An authorized program has directed that an address space be swapped out or in. A common use for the requested swap-in is to bring in an address space to cancel the current job, session, or started task.
- **Transition Swap**—MVS requires that V=R job steps or nonswappable programs be allocated frames in preferred, nonreconfigurable storage. These swaps may be frequent in logically partitioned systems, since even so prosaic an activity as a tape mount causes the requesting address space to become nonswappable. An IBM maintenance update reduced the frequency of such MOUNT-related swaps. A transition swap causes such an address space to be swapped out just before the V=R or nonswappable status is

fully established. When it is subsequently swapped into the appropriate kind of storage, the status is made effective.

- **Auxiliary Storage Shortage Swap**—Auxiliary storage must not be allowed to run out. When its usage exceeds a threshold as described in Appendix B, SRM directs that the address space acquiring frames at the highest rate be swapped out, and causes prominent operator messages to be displayed. At this time, address space creation is inhibited, and the system can rapidly "dry up." The condition is easily relieved by PAGEADDing an additional local page data set. When an installation experiences this condition, corrective action is easy to implement, but frequent occurrences need analysis and prevention.
- **Real Storage Shortage Swap**—Much like the previous case, this kind of swap is triggered when thresholds for central storage use are exceeded. When a threshold is exceeded, address space creation and swap-ins are inhibited, and directed swap-outs are initiated of swappable address spaces holding the largest number of frames. This condition is not as easy to relieve as the auxiliary storage shortage. Frequent occurrence (more often than once a week) of a real storage shortage condition may indicate exhaustion of the central storage capacity. The thresholds are OPT parameters (MCCFXEPR and MCCFXTPR); in systems with a high but stable level of page fixing, the thresholds may need to be adjusted.

Managing swapping in systems with significant batch and TSO workloads is challenging and often difficult.

A.6. Storage Isolation

Storage isolation is a means of altering the sometimes inappropriately democratic process of page stealing, designating some frames of some address spaces as exempt from stealing. It came into existence in an informal way in the MVS/SE1 time frame with a popular system modification or *ZAP* known at the time as *fencing*. An address space's working set could be explicitly defined in this way, designating a number of frames that would be exempt from page stealing ("fenced off") under normal circumstances.

In MVS/SE2, fencing was formalized and externalized as storage isolation. Four new IPS parameters were introduced, providing several useful control options.

A further use for storage isolation was found when *extended swap* was introduced, first in an Installed User Program (IUP) and later as part of MVS/SP 1.3. Because the minimum protected working set is preserved across swaps, specifying storage isolation for TSO address spaces above the true working set leads to "enrichment" of the swap-in working set (known as the *swap group*). The balance between swap paging delay and demand paging delay may be adjusted by varying storage isolation until an optimum point is found. The tradeoff is not linear, because swap paging is more efficient than demand paging. Adding five or ten pages to the swap group might increase swap-in delay by less than 20 milliseconds but prevent three or four page faults, each taking 30 to 50 milliseconds to resolve.

In systems with expanded storage, storage isolation is applied to the sum of frames in central and expanded storage, and does not inhibit page stealing to expanded storage. Such page stealing and subsequent page fault page-in is normally fast enough (20–50 *micro*seconds each way) to disregard. The main value of storage isolation in systems with expanded storage is to prevent migration of pages belonging to the address space from expanded to auxiliary storage.

A.6.1. Storage isolation parameters

Each storage isolation parameter is specified as a value pair, with the first value less than or equal to the second. The upper value for frame counts may be specified as "*" to denote the maximum available value: 32,767 in MVS/370 and MVS/XA, $2^{31} - 1$ in MVS/ESA. The parameters are:

- CWSS=(low,high)—specifying the range of protected frame counts for the common area, including PLPA, EPLPA, CSA, and ECSA. CWSS is a global IPS parameter, and must precede the first domain definition in the IPS.
- CPGRT=(low,high)—specifying the range of acceptable page-in rates for the common area, in auxiliary pages per real-time second. CPGRT is also a global IPS parameter, preceding domain definitions in the IPS.

- PWSS=(low,high)—specifying the range of protected frames for an address space. PWSS appears in the performance group period definition.

- PPGRTR=(low,high)—specifying the acceptable paging rate for an address space in auxiliary page-ins per resident second. PPGRTR appears in the performance group period definition. An obsolete but still accepted PPGRT parameter denotes page-ins per execution-time second for ordinary address spaces but was equivalent to PPGRTR for address spaces in cross-memory mode. PPGRT should not be used.

Each frame-count parameter interacts with its corresponding page-in rate parameter. The protected frame count at any time is called the *target.* The target starts at the low frame-count value and is increased in steps when the page-in rate exceeds the upper page-in rate value until the upper frame-count value is reached. The reverse takes place when the page-in rate is less than the lower value.

Considerations relating to storage isolation include:

- Specifying a *maximum* frame-count value implies that any frames held by an address space (or the common area in the case of CWSS) in excess of that count are *preferred* for page stealing. Therefore, the maximum should usually be "*" unless every address space in the performance group is nonessential. In the case of an "unloved" address space, a restrictive maximum PWSS will guarantee preferred page stealing, possibly shielding preferred address spaces from stealing. (This technique is sometimes called *negative storage isolation.*) The maximum value for CWSS should always be "*." Negative storage isolation for swappable address spaces should not be used since *any* storage isolation suppresses the operation of the Working Set Manager.

- Storage isolation does little good in systems with expanded storage, for workloads not explicitly excluded from expanded storage. It serves only to defer migration, not page stealing. Consequently, storage isolation in systems with expanded storage should be specified only for nonswappable address spaces (preferably as negative storage isolation on the unloved ones), and to define the swap

group sizes for TSO workloads barred from expanded storage.

- As in any regulated system, there should be enough of a neutral range between each set of upper and lower limits (as for page-in rates) to permit periods of stable operation without adjustment. A series of measurement experiments is the best way to determine appropriate ranges. A suggested starting point is to estimate the maximum value and then make the minimum about 80 percent of the maximum.

- Storage isolation for first-period TSO (and usually second-period as well) is not subject to page-in-rate-based adjustment, since the time in short transactions is insufficient to allow a page-in rate to be calculated. The form of storage isolation for TSO initial performance periods is usually specified as

```
PWSS=(144,*)
```

  where "144" is a typical value with IBM 3390 or RAMAC II or equivalent paging devices, providing some enhancement of the swap group size if the observed typical frame count for first-period transactions is about 140–150 frames. A short period of observation with a performance monitor can show the actual frame counts for TSO transactions at an installation.

TSO storage isolation is of limited value in systems with expanded storage unless the TSO workload is barred from expanded storage. In that situation, and in systems with no expanded storage at all, TSO storage isolation of response-critical performance periods is the closest thing to tuning magic available in MVS. A trivial increase in swap I/O time returns a many-fold reduction in page-in delay. Storage isolation for TSO is most critically needed when no swap data sets are in use (the preferred configuration) to ensure a minimum of single-page-movement I/O.

A.7. Dispatching Priorities

We have looked at SRM's role in controlling swapping and page stealing. Its other major function is the control of dispatching priorities. In a default MVS system, SRM does not do this. Spe-

cific actions must be taken to wrest control of dispatching priorities from the users and subsystem suppliers, and place it firmly with SRM.

Dispatching priority in MVS exists as 256 distinct levels, ranging from hexadecimal 00 to FF. Address spaces are associated with dispatching priorities. Global SRBs are dispatched above priority `X'FF'`. Without SRM control, an address space can be assigned a dispatching priority directly through the DPRTY JCL parameter or through a corresponding TSO LOGON option. A JCL example that might be found in the default JCL for JES2 is `DPRTY=(15,14)`, equal to `X'FE'`, a priority so high that virtually nothing can get a higher one. Allowing such individual, uncoordinated control for each batch job, started task, and TSO or APPC/MVS session is likely to make the operating system's behavior inconsistent if not unpredictable.

As early as the time of MVS/SE1, it was recognized that central control of dispatching priority was essential if MVS was to be a trustworthy and consistent production operating system. The first such control in MVS was a rather tentative automatic priority group (APG) covering a narrow subset of dispatching priorities. In SE2 and since, APG was superseded by the ability to give SRM control of all dispatching priorities, and to make 160 of the 256 possible priorities available to address spaces.

Dispatching algorithms have evolved as well. To avoid confusion, we shall consider only the capabilities as of MVS/SP 2.1.7 and later. These were not changed through MVS/ESA SP 4.2.2. Chapter 4 touches on new changes to dispatching algorithms in support of goal mode and references a CMG paper that gives full details.

A.7.1. Translation

The system used to denote dispatching priorities under SRM control is different from the simple (0–15,0–15) scale used in the JCL DPRTY parameter. Through Version 4, an IPS parameter called APGRNG specified the range of priorities to be "owned" by SRM. Each value may be in the range 0–15; the lower may equal the upper. To prevent address spaces from claiming the top priorities, the upper value should be 15. To allow the full range of 160 priorities (organized as ten groups of 16) to be under SRM control, the lower value must be no greater than 6. Two views were common for APGRNG: Minimalists argued for APGRNG=(6-15) or (5-15); advocates of full control and ease of interpretation favored (0-15).

In Version 5 compatibility mode, the dispute ends: the parameter is dropped and (0-15) is always used.

A.7.2. SRM dispatching priority groups

As specified in the IPS, dispatching priorities are organized in ten sets of 16 levels. Each set corresponds to the first hexadecimal digit of a priority designation. Within each set, there is a two-way division; the upper six levels of the set are individually specifiable as fixed priorities, while the lower ten levels in the set form a mean-time-to-wait group.

Dispatching priority is designated by the "DP" parameter or variations thereof. If a particular priority set is designated as "s," fixed priorities are specified as DP=F*s*4 (highest) through DP=F*s*0 to DP=F*s* (the lowest fixed priority),[12] and the mean-time-to-wait group of the set as DP=M*s*. The overall pattern looks like this:

SRM Priority	"Real" Priority	Rank
F94	X'9F'	Highest
F93	X'9E'	...
...	...	...
F90	X'9B'	...
F9	X'9A'	Lowest of the high
M9	X'90'–X'99'	Highest MTTW
...	...	...
F5	X'5A'	About in the middle
...	...	...
F04	X'0F'	Highest of the low
...	...	...
F00	X'0B'	...
F0	X'0A'	Lowest fixed priority
M0	X'00'–X'09'	Dead last

A.7.3. Algorithms

A fixed dispatching priority places an address space on the dispatching queue in a position relative to those of address spaces

12 This curious designation looks like an afterthought, and it is. An obsolete priority type called "rotate" occupied this slot previously. DP=R*s* was the old designation.

with higher or lower dispatching priorities. Within a given dispatching priority, address spaces are in a FIFO subqueue, ordered dynamically according to the sequence in which they most recently became dispatchable. (An inactive or nondispatchable address space is removed from this *true-ready* queue and added to the end of the subqueue for the specific priority when it again becomes dispatchable.)

The mean-time-to-wait (MTTW) algorithm divides each MTTW group into ten distinct fixed levels according to the average time between WAITs in each address space. Those with short intervals between WAITs (usually for I/O completion) will have higher priorities; those with long stretches of CPU use without intervening I/O activity have lower priorities. The priorities are reset frequently, at the interval of an SRM second, so a program with a changing pattern of CPU-to-I/O activity balance is managed properly at each stage in its execution. MTTW faithfully performs the "I/O-bound high, CPU-bound low" allocation of dispatching priorities that was so hard to manage in MVT, and does it automatically according to the changing demands of each address space.

All of this benefit of MTTW can be lost, however, if multiple address spaces are not free to compete with each other in a single MTTW group. Even so, MTTW may still fail if the wait time interval used by SRM to assign priorities is not well matched to the behavior of the address spaces. In Appendix B, we'll see how to use a performance monitor to observe MTTW operation and correct it if necessary.

A.7.4. Adjustment

SRM has several ways to adjust dispatching priority for an address space. For swappable address spaces, in which preference is to be given to short-running transactions, the progression through performance periods is the means of making significant changes in dispatching priority at each period change. A TSO performance group may start out with DP=F63 in the first period, move to DP=F61 in the second period, and then to DP=M5 in later periods.

For nonswappable workloads, there is no progression through performance periods, so adjustment is ordinarily limited to that available through MTTW or according to the fair access handling of fixed dispatching priorities or for each MTTW slot. If, for

instance, there are several CICS application-owning address spaces at the same fixed priority, they will tend to have equal effective priorities, although the address spaces with more frequent WAITs will tend to have lower effective priorities.

Suppose, however, that activity alone does not determine importance. In this case, the final SRM dispatching priority adjustment mechanism may be needed. That is *time slicing*, used to periodically alter the dispatching priorities of address spaces in one or more performance groups.

A.7.5. Time slicing

For systems with conflicting CPU-intensive workloads, time slicing may be used to apportion favored access to the CPU resource. Another common use of time slicing is to restrict the impact of address spaces, normally having high priority and widely varying CPU demands, on the rest of the system. Notorious examples include JES2, JES3, and DFSMShsm (HSM, IBM's Hierarchical Storage Manager).

Returning to our original view of time slicing, consider a system with two independent CICS subsystems, each serving a different group of users two time zones apart.[13] Each of them needs equal preference for the CPU for 40 percent of the time. Two performance groups of TSO users in the first period need such preference 10 percent of the time, and no workload is to be preferred the remaining 10 percent of the time.

With time slicing, we can designate a normal DP for each of the three competing workloads and a higher TSDP for the favored period. We also need to specify a time-slicing pattern, TSPTRN, to select each time-slice group for the appropriate number of time slices, in order. The skeleton IPS entries might look like this:

```
TSPTRN=(1,2,1,2,3,1,2,1,2,*) /* TIME SLICING PATTERN */
...
PGN=5,(TSGRP=1,DP=F62,TSDP=F63,...   /* CICS1 */
PGN=6,(TSGRP=2,DP=F62,TSDP=F63,...   /* CICS2 */
PGN=8,(TSGRP=3,DP=F62,TSDP=F63,... /* TSO1 */
PGN=10,(TSGRP=3,DP=F62,TSDP=F63,...  /* TSO2 */
PGN=14,(DP=F62,...   /* OTHER WORKLOAD (NO TSDP) */
```

The TSO address spaces will be preferred only once per ten SRM seconds and will compete equally for the CPU another SRM sec-

13 The time zone misalignment invalidates any scheme for statically distributing the CPU resource.

ond in ten. On fast systems this is no problem, but on a relatively slow CPU, such as an IBM ES/9000 9221-130, there may be a perceptible irregularity in TSO response time. To avoid this problem, the parameter

```
TUNIT=5
```

might be placed ahead of the TSPTRN line in the IPS, thus performing the time slicing five times per SRM second.

Time slicing is a special solution to a specific problem of otherwise unmanageable workloads. It has an overhead cost, particularly if TUNIT is used, and should be tried only after other approaches have failed.

A.7.6. Layout

A sample layout of dispatching priorities is shown in the following table.

DP	Name	Description
FF	*MASTER*	Predetermined priority
FF	GRS	Predetermined priority
FF	TRACE	Predetermined priority
FF	PCAUTH	Predetermined priority
FF	DUMPSRV	Predetermined priority
FF	CONSOLE	Predetermined priority
FF	ALLOCAS	Predetermined priority
FF	RASP	Predetermined priority
FF	XCFAS	Predetermined priority
FF	SMF	Predetermined priority
FF	IOSAS	Predetermined priority
FF	CATALOG	Predetermined priority
F94		Reserved for emergencies
F92		Real-time monitor (not VTAM session)
F84		Historical monitor
F82	VTAM	Network manager
F80		VTAM monitor
M8		System address spaces (SMS, ASCH, etc.)
F74		DB2 subsystem monitor

DP	Name	Description
F72		DB2 IRLM
F71		CICS monitor
F70		Primary CICS terminal-owning region
F64		DB2 MSTR
F62		DB2 DBM1
F60	LLA,VLF	LLA and VLF housekeeping
F6		Hot APPC/MVS transactions
F54		Primary TSO, period 1
F52		Secondary TSO, period 1
F51		APPC/MVS E-mail transactions
F50		DB2 DIST
F44	JES2	JES2 TSDP
F43		CICS application/resource regions
F42	DFSMShsm	HSM TSDP
F40	RMF	RMF writer address space
F34		Primary TSO, period 2
F33		Default TSO, period 1
F30	JES2	JES2 base priority
M3		Primary TSO, period 3
F24		Secondary CICS TOR
F22		Secondary CICS other address spaces
M2		Secondary TSO, period 3
M2		Default TSO, period 2
M2		Preferred batch
M2		Production batch, period 1
M1		Last-period TSO/batch, default STCs
F04		Configuration/change monitor
F02	DFHSM	DFHSM base priority
F0		"Bottom feeder" problem jobs
M0		"Sponges" to measure CPU reserve

Creating such a layout helps to ensure that SRM (and the person in charge of directing SRM) is in control of dispatching priorities and that the priorities in the IPS reflect the installation's workload priorities. In creating a dispatching priority layout,

the suggestions made in Chapter 5 relating to the ideal of a "well-ordered CPU" should be reviewed. Remember that certain system address spaces have predetermined dispatching priorities, and that server address spaces usually need to be above their users in priority. Most monitors need to rank above the workloads they scrutinize to give accurate measurements of utilizations and delays. In the sample, the MVS/ESA address spaces whose priorities are predetermined are shown in order. We assume that the system supports production TSO, CICS (with DB2), and a somewhat less important IMS service. JES2 and HSM are time-sliced to ensure that response-critical address spaces may be dispatched with minimum delay. Setting up an order of priority for access to the CPU may also be helpful in establishing a similar ordering for access to expanded storage. In other words, it makes little sense for an address space favored for CPU access to be paged and swapped to and from auxiliary storage. In Appendix B, we'll put all these considerations together.

A.8. Summary

We have introduced numerous SRM terms and, in passing, some representative parameters as well. With this preparation, we can move on to Appendix B, presenting SRM parameters in a more structured manner.

A.9. Questions

1. Examine the RMF Workload Activity report or an equivalent source to determine the percentage TSO transaction completion distribution by performance period in your system. Is the pattern what you expected? What changes would you make?
2. Using whatever tools you have available, obtain a transaction-level sample of a busy TSO hour. GTF SRM trace is one way to obtain the necessary data. You should create a tabulated list of service units for completed transactions. Using a tool such SAS PROC UNIVARIATE, create a cumulative frequency distribution plot or table. You may now look for the natural break points or flat spots in the distribution. How do they compare with the performance period definitions? What needs to be done to reconcile the natural

workload pattern with the way you've been managing the workload?

3. Examine the domain constraints in your MVS system. How many address spaces will be in the multiprogramming set if all domains have at least as many address spaces ready as their minMPLs? How does that number compare with the maximum MPL through the busiest hour for which you have RMF data? Should your minMPLs be lower?
4. Again looking at the domain constraints, do the maxMPLs restrict the system from becoming fully utilized when only a single workload is active? Is such a possibility likely? How should you change the domain definitions to allow full utilization in any workload distribution while making sure that important workloads are favored? Use the full capabilities of ASRV and DSRV to define your policy to SRM.
5. Do your dispatching priorities reflect both the intrinsic relationships between workloads and the relative importance of each workload to the enterprise? What adjustments are needed?

Appendix

B

SRM (Compatibility Mode) Parameters

B.1. OPT Parameters

Successful MVS performance management entails far more than merely selecting the right values for SRM controls. Numerous other PARMLIB parameters are important in determining how a system runs. The physical configuration and its interaction with the workload is still the key fundamental factor influencing system performance and throughput. But no matter how much effort goes into perfecting the system's configuration and set-up, applications and application programs designed without an appreciation of performance factors can undo it all.

On the other hand, the SRM parameters remain very accessible and can have a significant effect on system performance, workload performance, and throughput, even to the extent of compensating for configuration and application deficiencies. For that reason, but always with the awareness that we're looking at only part of the picture, we now look at the SRM controls in detail.

The IEAOPTxx (OPT) member of SYS1.PARMLIB defines global controls for SRM. In general, OPT parameters affect SRM algorithms directly, and workloads indirectly, through the operation of the algorithms. To set OPT parameters correctly is not merely a tuning action in response to some crisis. It is a necessary part of reconciling the system resources to the needs of the workloads. We'll consider these parameters in logical groupings,

in approximate order of importance. Extended discussions of related performance management actions are inserted along with the discussion of the parameters involved.

B.1.1. Expanded storage selection criteria

Expanded storage, when present or defined in the configuration, is a key critical resource in MVS/ESA systems. New capabilities as of MVS/ESA SP 4.2 facilitated precise management of the paging use of this resource.

Pages are selected for move-out to expanded storage based on the interaction of the expanded storage migration age, and in some cases the system-wide high UIC, with a set of criteria in the *Criteria Age Table* (*CAT*). There are (as of SP 4.2) seven different categories of pages enumerated in the CAT; six of them are further divided into workload categories specified with FORTRAN-style subscripts in the OPT.

The default subscript values, carried over with some minor changes from pre-SP 4.2 releases, are:

0—in paging decisions, nonswappable or privileged address spaces, or common-area pages; privileged address spaces only in swapping and expanded storage decisions.

2—in paging decisions, APPC/MVS or TSO address spaces; in swapping and expanded storage decisions, TSO or APPC/MVS sessions in long, detected, or terminal waits.

1—for pages of address spaces in neither category 0 nor category 2— usually batch.

SP 4.2 introduced the capability to specify additional subscripts, dropped some parameters that proved to be unnecessary, and added VIO and hiperspace pages to the candidates for management by workload category. The subscripts (or index values) 3–99 are specified on the ESCT*xxxx* parameters in the OPT. They are linked to workloads with the ESCRTABX parameter on the DMN statement[s] in the IPS. Address spaces in domains for which the index value has not been specified will be covered by the defaults shown above, according to address space classification. If a partial set of parameters is specified for an index value, the missing ones are filled in by the defaults. On the other hand, if an index is specified in the IPS that is not defined at all in the OPT, all pages in that domain will be treated as if an index of 1 had been specified.

A page will be sent to expanded storage if the current migration age (for expanded storage as a whole) exceeds the address space's (or common area's) criteria [*sic*][1] age for the class of page being considered. For swap trim and non-terminal-wait swap working-set pages, the migration age is augmented by the current system-wide high UIC. For terminal-wait working-set pages, the UIC correction applies, and the address space's think time is added to the criteria age as well.

The defaults for the initial set of expanded storage controls appeared to have been selected before appreciable production experience had been acquired. When migration age got down to the maximum criteria age in the default CAT, 100 seconds, migration might already have reached disruptively high levels. When the ESCTVIO and ESCTBDS controls were added in enhancements to MVS/SP 2.2.0 and MVS/SP 3.1.0, respectively, they were both set at a more aggressive default of 900 seconds. The defaults finally were realigned in MVS/ESA SP 4.2 to significantly higher values.

The expanded storage controls are described in Chapter 3 of *Initialization and Tuning Guide*, and enumerated in *Initialization and Tuning Reference*. The default settings are appropriate for systems with adequate expanded storage. (In such a system, migration age is rarely less than 1000 seconds.) Following a discussion of general approach below, we'll look at each parameter, its default settings, and suggested changes from defaults.

Setting up Criteria Age Table entries

Even if you have adequate expanded storage for today's needs, there may come a time when you don't. It's necessary to create a "triage" plan to determine who wins and who loses when the demand exceeds the supply.

The easiest way to make an initial difference is to focus on two "magic numbers," 32,766 and 32,767. The first one is the "last choice" designation. If, for instance, you define

```
ESCTSWWS(13)=32766
```

then whatever address spaces pass through a performance group to a domain with **ESCRTABX=13** specified will be swapped to expanded storage only when the migration age exceeds 32,766 seconds, or when the free availability of frames allows migration

1 The term should be in the singular, but IBM has "legitimized" this deviation from proper usage.

age to be disregarded. Migration age is ignored whenever the number of available expanded storage frames exceeds a threshold that is several (four to five) times the value of MCCAECOK, a parameter described on page 348.

To bar a class of pages from *ever* going to expanded storage, use the second magic number, 32,767. Since the migration age can never exceed this value and the exception for available frames is not considered in this one case, the pages will go only to auxiliary storage. Remember, a page goes to expanded storage only if the current migration age is higher than the applicable criteria age. The basic policy is modified by examining the UIC (in the case of page steals) and the last think time for the address space (in the case of swap-outs). For other than swaps based on resource shortages, transition swaps, and request swaps, pages are not moved to expanded storage unless the central storage frames are needed to replenish the available frame queue.

Once you've ruled out the workloads to be barred, set priorities for assignment of other pages to expanded storage. Determine the approximate critical migration age[2] of your system. Plot the migration rate against the migration age through a varied sam-

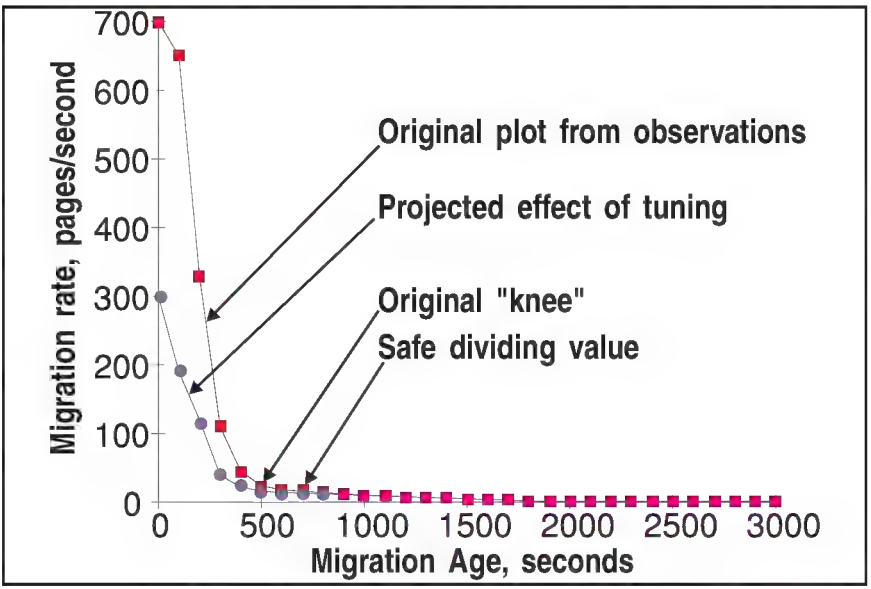


Figure B-1. Critical Migration Age and Effect of Tuning.

2 *Critical migration age* is a term invented by the author.

ple of observations, as shown in Figure B-1. The critical migration age corresponds to the "knee of the curve." There should be a knee. The criteria ages for workloads that are not performance-critical should be set well above the critical migration age. Other workloads can have lower criteria ages in inverse order of importance. (The more important the workload segment, the lower the criteria age can and should be.) The defaults will be instructive in setting these levels.

Now add ESCRTABX parameters to the appropriate domains to activate the criteria age settings. The system should then be measured over a representative peak period. The effect of establishing selective criteria table entries should be to move the knee of the curve to the left (as seen on the graph), to make the knee less distinct, and to limit the maximum migration rate. Make any criteria age table adjustments necessary to secure this result. Continue measuring and adjusting until there is an acceptable balance between workload response time and migration rate under all load conditions.

B.1.2. Expanded storage criteria parameters

Each of the parameters described in this section may assume a value from 0 to 32,767. In each case, a value of 32,767 bars the corresponding page from eligibility for movement to expanded storage. Each parameter other than ESCTVF, ESCTBDS, and ESCTVIO requires a subscript or index value ranging from 0–99 and enclosed in parentheses. For ESCTBDS and ESCTVIO, the index is optional and in the range 3–99. By using index values of 3–99, criteria ages may be set to deal with specific workload management needs. It's prudent to leave the default values "as is" until a compelling reason to change a default becomes evident.

ESCTSTC, stolen pages

Pages stolen from active address spaces are likely to be needed again a short time later. Thus, this category should have the lowest criteria age for each workload class. The defaults are in line with this suggestion:

```
ESCTSTC(0)=100
ESCTSTC(1)=250
ESCTSTC(2)=250
```

A page of this class will go to expanded storage when the following relationship is true:

migration age> criteria age

ESCTSWTC, swap trim pages

Pages stolen from address spaces just before swap-out are less likely than ordinary stolen pages to be needed again after swap-in. The defaults reflect the difference:

```
ESCTSWTC(0)=450
ESCTSWTC(1)=450
ESCTSWTC(2)=350
```

A page of this class will go to expanded storage when the following slightly more complex relationship is true:

UIC + migration age > criteria age

ESCTSWWS, swap working set

The pages moved out to expanded storage in a swap-out have, in general, already been passed over for logical swap and have aged for some time in central storage until the frames are needed. In many cases, particularly for numerically intensive workloads with large working sets, swapping to auxiliary storage may be an acceptable or even preferable alternative. Especially for batch jobs, the I/O delay of a swap-in from AUX is fully overlapped with other processing and the effective CPU path length may be shorter than that for a swap-in from expanded storage. The defaults in OS/390 Release 1 are:

```
ESCTSWWS(0)=450
ESCTSWWS(1)=450
ESCTSWWS(2)=350
```

Selection for movement to expanded storage for this class of page depends on the truth of the relationship:

UIC + migration age - think time> criteria age

The think-time term drops out for nonwait-state swaps, such as exchange and unilateral swaps.

It may be advisable to increase the value for ESCTSWWS(1) to 3600 (one hour) or more if expanded storage is in short supply, especially if central storage is also limited. Doing so will ensure that unilateral swaps of nonproduction batch jobs and late-period TSO go to expanded storage only if migration is very unlikely.

ESCTPOC, explicitly paged-out pages

Pages enter this category as the target of an explicit PAGEOUT request to MVS. A program doing this kind of explicit real storage management has "decided" that the page is not currently needed, and the defaults support this position:

```
ESCTPOC(0)=1200
ESCTPOC(1)=1200
ESCTPOC(2)=1200
```

Selection for movement to expanded storage for this class of page is according to the truth of the same simple relationship that applies to stolen pages:

migration age> criteria age

ESCTBDS, hiperspace pages

When this parameter is specified without a subscript, it sets a global default for all hiperspace pages in the system. Pages of ESO hiperspaces are not migrated but "cast out," later causing "misses" when access is attempted. The address space owning such a hiperspace needs to provide recovery for these misses, retrieving the data from another source, usually DASD.

When a subscript in the range 3–99 is added, the parameter defines an alternative criteria age that may be associated with the address spaces of one or more domains. The default value for ESCTBDS is 1500 seconds, and the eligibility relationship is the same as for stolen pages. Setting the criteria age to a lower value for essential workloads such as production CICS may ensure effective use of expanded storage for such applications as hiperspace buffering for VSAM local shared resources.

ESCTVIO, VIO swap-out pages

This parameter is similar in all respects to ESCTBDS, except that it applies to Virtual I/O (VIO) pages. It has the same default value of 1500 seconds.

ESCTVF, Virtual Fetch pages

Virtual Fetch is a system service used to improve program fetch time for programs, particularly non-reenterable ones. It is, to date, used only on behalf of IMS/VS or IMS/ESA. For systems on which IMS is a principal workload, the low default criteria age of 100 ensures responsive Virtual Fetch performance. Where IMS is a secondary workload, but Virtual Fetch is used (perhaps

for compatibility with another system), the value for this parameter should be increased in inverse proportion to the value of the workload. No subscript is allowed for this parameter; all Virtual Fetch pages are affected equally.

Real Storage Frame Thresholds

The two parameters presented in this section, both added in MVS/ESA SP 4.2, "externalize" the thresholds the SRM uses to manage the available frame queues in expanded storage and central storage, respectively. The low ("LO") value of each parameter translates to the threshold at which migration or stealing commences; the upper threshold is the "OK" value at which frame replacement stops.

The thresholds are not rigidly fixed. In systems without expanded storage or with substantial swap activity from auxiliary storage, MCCAFCOK (the high threshold of MCCAFCTH) is adjusted before a swap-in to ensure that page stealing secures enough frames to accommodate the entire swap group plus a buffer above the current value of MCCAFCLO. Various other factors cause adjustment of both of the LO and OK thresholds as system requirements dictate.

MCCAECTH, expanded storage frame thresholds

MCCAECTH specifies the starting point for management of the expanded storage available frame queue. It is specified as a value pair, with the second value (MCCAECOK) not less than the first (MCCAECLO), and both in the range 1–32,767. The default is (150,300).

When the number of free frames falls below MCCAECLO, both cast-out of discardable hiperspace pages and migration proceed until MCCAECOK frames are available. The targets are adjusted dynamically to anticipate unusual demands and prevent immediate re-invocation of migration.

Migration may be viewed as a progression around a circular queue of all expanded storage frames, started and stopped as the thresholds dictate, but always resuming at the frame just beyond the last stopping point. In a simplistic view of the process, each frame is identified as "new" or "not new." When the migration routine reaches a "new" frame, it removes the attribute and passes over the frame. The next time the frame is reached, it is recognized as not new and thus eligible for migration, unless protected by stor-

age isolation or covered by some other exception condition. *Migration age* is the time needed to reach the average frame a second time. This simple process is in contrast with page stealing, in which UIC provides a quantitative measure of page age. There is no UIC for expanded storage frames.

Migration is considerably more complex owing to the associations of swap groups and other blocks of frames, but the picture described above is a useful approximation.

You may wish to change the thresholds for the expanded storage frame queue to deal with some special need. If you do, make sure to maintain a substantial gap (the default is 150 frames) between the high and low thresholds. Not doing so may lead to increased overhead as the migration routine is invoked more frequently.

MCCAFCTH, central storage frame thresholds

MCCAFCTH specifies the starting point for management of the central storage available frame queue. It is specified as a value pair, with the second value not less than the first, and both in the range 1–32,767. The default is (150,300).

When the number of free frames falls below MCCAFCLO, a page stealing cycle commences and proceeds until MCCAFCOK is reached. The targets are adjusted dynamically to anticipate unusual demands such as large swap-ins and prevent immediate re-invocation of the stealing process.

B.1.3. Central storage management details

In the highly integrated central storage management environment of MVS/ESA SP 4.2 and its successor releases, it's necessary to go back to the demand factors to get a good understanding of page assignment, reassignment, and swapping. The two principal demand factors are page faults and swap-ins.

When a page fault occurs, page movement or a page-in is not needed for a first-time fault on a newly obtained page. Otherwise, the page fault requires both that the page be moved from its current location and that one or more frames be assigned for it. Since all page-ins from auxiliary storage and some move-ins from expanded storage are blocked, the need for multiple frames is common.

In much the same way, swap-ins may or may not require additional frames. Processor storage swaps (including logical swaps) in central storage need no frames. A swap-in from expanded

storage needs enough frames for the primary working set, while a swap-in from auxiliary needs frames for the full swap group.

Where do the frames come from? The simplest case is to take the frame or frames (in the case of a blocked page-in) needed to resolve a page fault from the address space itself. This occurs if the Working Set Manager is managing the address space, which at the moment holds more frames than its managed target. In more ordinary cases, frames come from the available frame queue.

When the AVQLOW condition is signaled to SRM by RSM (Real Storage Manager), a process to replenish the frames begins. In addition to the ordinary case of the frame count going below MCCAFCLO, RSM will call for AFQ replenishment in anticipation of a large demand for frames, such as a swap-in. The process has several mutually exclusive choices. In the order listed below, each choice is more disruptive to system operation and impedes workloads more than the preceding one.

In each case, the process stops as soon as the current OK threshold (which may be adjusted based on the number of frames needed) is reached or passed. Pages stolen from an address space, at equal UICs and from adjacent virtual pages, are grouped in blocks in anticipation of blocked page-in. Details of the algorithm are not relevant, are proprietary, and are subject to change in any case. The order of selection is not authoritative but is assumed based on analysis, IBM presentations, and the observations of colleagues. Not too much significance should be put on the placement of a given step. The essential point of the discussion is the progression from clearly surplus frames to those whose loss can cause workload delays.

- If the system-wide average high UIC is more than about 200, page stealing is the choice, following these steps:
  - The Working Set Manager may have established steal targets for address spaces it is managing. The frames that can be safely stolen are taken first.
  - Storage-isolated address spaces with central storage frames above their maximum PWSS values are preferred for stealing next. The full excess is taken from each address space in turn, stopping when the steal request is satisfied.

- The system-wide high UIC is set as the starting point *steal criterion* for ordinary page stealing. The steal routine visits each address space in order and then concludes with the common area. For each address space having at least one frame at or above[3] the current steal criterion, up to four or ten frames at or above the criterion UIC are taken. More frames are taken when the current steal criterion UIC is high; fewer when it is low.

  The number of frames taken from an address space may be increased beyond the four or ten. If a single address space has a larger number of frames at the current steal criterion, and they are at sequentially contiguous virtual addresses, they are all taken, even going beyond the number needed to complete the current steal requirement. The pages related by common UIC and adjacent virtual addresses constitute the unit of implicitly blocked page stealing. When the number of such pages is greater than the maximum burst size[4] on the paging device, multiple bursts are written in parallel by ASM, making this kind of page stealing look very much like a partial swap-out. If the stealing goes to expanded storage, the blocking information is retained through migration so that block faulting from auxiliary storage can be effective. Efficiency and parallelism are the keys to retaining acceptable system performance while accommodating numerically intensive workloads.

  When a page fault causes a page in the block to be brought in, all succeeding pages in the block are eligible to be brought in, up to the current AFQ count. For best operation of the blocked page fault mechanism, a sufficiently large number of free frames must be available, so the values of MCCAFCTH may have to be increased. If the application program has declared a page reference pattern for

3 The initial pass of ordinary page stealing may take frames above the criterion from storage-isolated address spaces. The UICs of such address spaces are not included in the system-wide high UIC calculation, and the UICs of frames in storage-isolated address spaces tend to be higher than those in the rest of the system. Ordinary page stealing is allowed in storage-isolated address spaces with frame counts above the PWSS minimum but below the maximum.

4 The burst size is the same as the "swap set" size for local page data sets: 30 slots on 3380s or 9345s, 36 slots on 3390s.

explicit block paging, the pages may be at discontiguous addresses and the entire block is eligible for block faulting, regardless of which page incurred the fault.

- After all address spaces are considered for stealing at the current criterion UIC, the common area is treated the same way.
- If the required number of frames has not yet been obtained after a complete pass, the criterion is reduced by one and the process repeated for address spaces and the common area. The looping behavior continues until the steal request is satisfied. Since the stealing began at a high UIC, which is usually correlated with a large number of eligible frames, the dire consequences of stealing down to zero UIC are unlikely. They are discussed below in connection with page stealing when UIC starts off at a low value.

- The next frames to be taken are those from address spaces (one at a time) designated for processor storage swap but not yet moved out to expanded storage. The few unchanged frames in this situation are immediately moved to the available frame queue, and the changed pages are moved in blocks to expanded storage, stopping when the steal request is satisfied. The address space is not swapped unless frames are needed after it is trimmed down to its strict working set or to the number of frames specified by MCCMAXSW in the OPT.

  Within this group of address spaces, those swapped for purposes of MPL adjustment (unilateral, ENQueue exchange, and exchange swaps) are selected first. They have been removed from the multiprogramming set, while "logically swapped" address spaces are eligible for immediate re-entry to the multiprogramming set.

- If the AFQ is not yet fully replenished, the page-stealing procedure described in the first step is now executed, but now it is more likely to continue until the pass at UIC=0 is unsuccessful, signifying that all remaining frames are V=R, fixed, storage-isolated, or just referenced since the last UIC update.

- The system is now in an emergency page-stealing condition. The entire ordinary page-stealing loop begins again, with the criterion once again set to the system-wide high UIC. This time, however, storage isolation of both address spaces and the common area is ignored.
- If the required number of frames cannot be obtained, domain MPLs may be set to their minimums and unilateral swap-outs commenced. Swap-outs of address spaces in domains at or below their in-targets may be initiated. At this point, the system is severely degraded, and will be perceived by interactive users to be unresponsive.

Page stealing satisfied by any of the first few steps is unobtrusive and not likely to lead to significant subsequent page-in delay. Stealing at low UICs will adversely affect the performance of the most recently visited address spaces. If the cycle ever proceeds to the point of emergency page stealing, mission-critical workloads may be adversely affected.

Other mechanisms exist in MVS to preclude the need for emergency page stealing. There are two MPL adjustment parameters in the OPT discussed in the following pages—RCCFXET and RCCFXTT—whose default values are set so as to provide a reserve of frames that are neither fixed nor involved in paging operations. Any upward adjustment of these parameters (for instance, in systems with very large central storage) should be done only in conjunction with careful measurements of page stealing frequency, to avoid triggering damaging and inefficient page stealing.

The OPT parameters MCCFXEPR and MCCFXTPR, also described later in this chapter, supply higher thresholds for the percentage of frames in the system that are fixed or allocated for paging. Exceeding either or both of these thresholds will trigger SRM actions to prevent exhaustion of the available frame queue. The ensuing Pageable Storage Shortage swap-outs, with their accompanying ominous messages, may serve to forestall gross degradation of major workloads.

Given the capabilities of MVS/ESA since SP 4.2 to deal with swappable workloads having very large storage needs, it may be advisable to increase the MCCAFCTH thresholds if such workloads are present. If, for instance, the maximum swap packet size is set at 3 megabytes (768 frames) and more than one such

job is likely to be running, page stealing should commence when the AFC falls below this value and should not stop until the likely block page-in requirement can be met without another page stealing cycle. In such a case, the specification might be

```
MCCAFCTH=(768,1000)
```

As with all such recommendations, a period of measurement and evaluation both before and after the change is needed to refine the setting.

B.1.4. Swapping adjustments

In SP 4.2, two new controls were introduced to tune the swapping process. The first can be used to limit the disruptive effect of single swaps, while the second regulates the cumulative effect of exchange swapping.

MCCMAXSW, maximum swap packet size

(*Author's Note*: IBM calls this parameter *swap set size*. That name is already used to describe the fixed 12-slot unit of swap data set allocation and use. It is also used loosely to denote the maximum contiguous-slot group size used by the Auxiliary Storage Manager [ASM], typically 36 slots [three tracks] on a 3390. I therefore coined the term *swap packet* to describe the maximum number of pages moved at each stage in a multistage swap.)

Suppose that a moderately large batch job has a trimmed swap group size of 830 pages, about 3.4 megabytes. If swapping is done to expanded storage at a streaming rate of 30 microseconds per page, about 25 milliseconds of CPU time will be diverted to page movement for the swap-out. The swap-in will be broken up into the primary working set, swapped in together, and the secondary working set, brought in as needed. If the working set after swap-in resembles that before swap-out, perhaps 800 pages will be brought in as page faults, at about 50 microseconds apiece. The swap-in thus consumes about 40 milliseconds, for a total of much less than a hundredth of a second of pure CPU as overhead for each swap sequence.[5] This is not yet a severe load on the system.

The same job swapped to auxiliary storage requires 23 dedicated 3390, 9345, or RAMAC local page data sets on 23 channel and stor-

5 A swap sequence is simply the combination of a swap-out and the ensuing swap-in.

age director paths for full concurrency, giving a swap-in time of about 40–50 milliseconds. A more realistic configuration of six local page data sets will give a swap-in time of 200–250 milliseconds, or 400–500 milliseconds for the swap sequence, with about the same CPU cost as for the expanded storage swap.

Now imagine a batch job ten times as big, with a trimmed working set of 8300 pages. Such jobs are commonplace in many engineering and scientific application areas. Now the swapping impact, if incurred all at once, can be very disruptive—if 8300 expanded storage frames are available, almost a tenth of a second of CPU time is consumed for a swap sequence. If swapping is to and from auxiliary storage, a swap sequence time of 6 to 7 seconds is incurred.

As of MVS/ESA SP 4.2, the intense impact of the swap sequence is spread across a longer time interval based on the value of MCCMAXSW. Successive trim packets of up to that number of pages are sent out as blocks of pages, followed ultimately by the final packet as the swap-out. Raising MCCMAXSW from its default value of 512 pages can make effective use of a well-matched paging configuration to deal with the swapping needs of very large swappable workloads.

Before changing this parameter, study the swapping statistics of your system carefully. Ensure that the paging subsystem (or expanded storage) can handle the anticipated load, and measure carefully before and after the change. What should happen is that the very large address spaces will not suffer excessive elongation, while the throughput of the system (as measured by total CPU and I/O service units) will increase. If the response times of interactive workloads have been suffering, they should improve.

The approach suggested here is necessary in any tuning exercise:

- Identify a problem to be solved.
- Secure appropriate measurements and analyze the problem to determine what kind of change might produce improvement.
- Predict what the expected effect of the change is likely to be; also seek out and predict possible negative consequences.
- Make a set of baseline measurements for later comparison.
- Implement the indicated change (one change at a time).

- Measure after each change.
- Assess the effect of the change on the direct area of concern, as well as on all possible affected peripheral areas. Test your theory of what the change should have done, using the measurements.
- Continue making changes to refine the original guess until a point of diminishing return is reached.

SWAPRSF, exchange swapping bias

Exchange swapping was the pariah of workload management tools throughout the history of MVS. However, it was implemented to meet a simple need: ensure that in a domain with limited MPL, a shorter-running job would not get trapped behind a longer-running one. Doing this is a simple way to ensure consistent batch turnaround times regardless of arrival order, thus contributing to customer satisfaction.

What, then, is so bad about exchange swapping? The simple explanation is that it was viewed in the early days of MVS as "unnecessary" compared with such obviously more worthy swaps as unilateral swaps for MPL adjustment and swaps of idle TSO users. Because all swapping in early MVS systems was an unqualified disaster, unnecessary swaps had to be eliminated.

Soon after exchange swaps had been "conquered," MVS swapping became much less of a problem. Swap data sets came first, reducing the impact of swapping on workloads sensitive to page-in delay. Logical swapping was next, eliminating many TSO terminal-wait swaps. Later, extended swap and the contiguous-slot allocation algorithm made swap data sets obsolete and, with storage isolation, made efficient swap paging with reduced page faulting possible. A big improvement came with the introduction of expanded storage in the IBM 3090, as well as the general trend to larger central storage sizes.

By the time MVS/ESA became available, the swapping burden had become insignificant for address spaces of ordinary size. It was time once again to consider the benefits of exchange swapping. However, prior to SP 4.2, doing so would have required reworking of thousands of IPSs constructed when exchange swapping was out of favor. With SP 4.2, the parameters and mechanisms arrayed against exchange swapping were all swept away.

One parameter was added to set a global bias for or against exchange swapping.

SWAPRSF has a default value of 10, corresponding roughly to the old default ISV of 100,000. It may be set as low as 0.1, effectively allowing exchange swaps at will, or as high as 100, making exchange swaps about one-tenth as likely as in the default case.

In systems with a substantial load of unscheduled batch having no sequential dependencies and widely varying run times, setting SWAPRSF to a lower-than-default value, perhaps to 5, will facilitate exchange swapping, improving uniformity of job completion durations. With a heavier batch load, including some moderately large jobs, SWAPRSF should be left initially at its default setting. When the dominant workload is NIC (Numerically Intensive Computing) batch, SWAPRSF might be increased to suppress exchange swaps and allow the "swapped out too long" swap reason to bring in such address spaces at 10-minute intervals.

B.1.5. Real storage management example

This example will show the relationships among expanded storage criteria table entries, workloads, and the parameters that control page stealing, migration, swap packet size, and exchange swapping frequency.

Let us assume that the critical migration age for the system is about 800 seconds, and that there are two CICS workloads (major production and pilot production), two TSO workloads (support and application development), and two batch workloads ("normal" and "heavy engineering"). The heavy engineering workload has typical working-set sizes of 30 megabytes. There are eight local page data sets on 9345B12s, each on separate paths. Therefore, 288 pages may be moved concurrently. Allowing for two data transfers per swap packet, an MCCMAXSW value of 576 fits the packet size to the paging configuration, and the default value of 512 is not far wrong.

The order of importance for these workloads is:

major production CICS
support TSO
heavy engineering batch
pilot production CICS

application development TSO
normal production batch
default (open shop) batch

There is not enough expanded storage to accommodate the entire workload. Heavy engineering batch is to be swapped to auxiliary storage. Other paging categories will compete for expanded storage in priority order. OPT settings to manage these competing workloads might be:

- global parameters

```
MCCMAXSW=576                    /*FIT SWAP PACKET TO AUX*/
SWAPRSF=20                /*MILDLY RETARD EXCHANGE SWAP*/
MCCAECTH=(150,300)                      /*DEFAULTS OK*/
MCCAFCTH=(600,900)                /*BIG SWAPS LIKELY*/
```

- criteria ages for default batch

```
ESCTSWWS(1)=32766               /*SWAPS TO E.S. LAST*/
ESCTSWTC(1)=32766               /*TRIMS TO E.S. LAST*/
ESCTSTC(1)=1000
ESCTPOC(1)=5000
ESCTVIO(1)=5000
ESCTBDS(1)=2000
```

- criteria ages for normal production batch

```
ESCTSWWS(5)=2000
ESCTSWTC(5)=2000
ESCTSTC(5)=1000
ESCTPOC(5)=2000
ESCTVIO(5)=1500
ESCTBDS(5)=1500             /*FACILITATE HIPERBATCH*/
```

- criteria ages for heavy engineering batch

```
ESCTSWWS(10)=32767                /*NO SWAP TO E.S.*/
ESCTSWTC(10)=32767                /*NO TRIM TO E.S.*/
ESCTSTC(10)=900
ESCTPOC(10)=900
ESCTVIO(10)=32766               /*VIO NOT EXPECTED*/
ESCTBDS(10)=32766             /*HSPACE NOT EXPECTED*/
```

- criteria ages for major CICS

```
ESCTSWWS(15)=32766          /*NORMALLY NONSWAPPABLE*/
ESCTSWTC(15)=32766          /*NORMALLY NONSWAPPABLE*/
ESCTSTC(15)=20
```

```
ESCTPOC(15)=50
ESCTVIO(15)=2000                    /*VIO NOT EXPECTED*/
ESCTBDS(15)=100                     /*ALLOW HSPACE LSR*/
```

- criteria ages for pilot CICS

```
ESCTSWWS(20)=32767              /*NORMALLY NONSWAPPABLE*/
ESCTSWTC(20)=32767              /*NORMALLY NONSWAPPABLE*/
ESCTSTC(20)=150
ESCTPOC(20)=250
ESCTVIO(20)=3000
ESCTBDS(20)=1500               /*ALLOW SOME HSPACE LSR*/
```

- criteria ages for support TSO

```
ESCTSWWS(25)=100
ESCTSWTC(25)=100
ESCTSTC(25)=30
ESCTPOC(25)=100
ESCTVIO(25)=700
ESCTBDS(25)=1000
```

- criteria ages for application development TSO

```
ESCTSWWS(30)=200
ESCTSWTC(30)=200
ESCTSTC(30)=150
ESCTPOC(30)=200
ESCTVIO(30)=1500
ESCTBDS(30)=3000
```

This example will be revisited later in this appendix, showing the corresponding IPS settings in context with the OPT settings.

B.1.6. MPL adjustment

MPL adjustment might be thought of as unimportant in systems with expanded storage. After all, most swaps start off as processor storage swaps, and paging delay from auxiliary storage is insignificant. However, a major change occurred in MVS/ESA SP 4.2, invalidating these assumptions. In today's systems it is possible to create two distinct workload categories, one that will ordinarily use expanded storage and one that will not. MPL adjustment and considerations relating to auxiliary storage paging and swapping apply, much as they

did before, to the part of the workload excluded from expanded storage.

IBM's *Initialization and Tuning Guide* discusses at some length the operation of the algorithm controlling system-wide MPL. The basic process requires that the value of each MPL adjustment variable be beyond the favorable limit of the corresponding control to increase MPL, while a single variable on the wrong side of the range can cause MPL decrease.

Default values for many MPL adjustment parameters have in the past been inappropriate, tending to reduce MPL before the CPU is fully utilized, yet allowing auxiliary storage paging delay to rise to harmful values on some systems without reducing MPL. Improved defaults were supplied starting with MVS/SP 3.1.3.

Ideally, MPL should be controlled so as to keep demand page-in delay of key workloads and CPU delay arising from expanded storage page movement to acceptable values. Unfortunately, there was no MPL control parameter based on paging delay or CPU delay, and none is workload-specific. In pre-SP 4.2 systems with expanded storage, MPL is not adjusted in response to page movement rates or migration activity. Lacking such ideal controls, less direct indicators of delay had to be used in the OPT. These included the page fault rate, the storage reference age (UIC), the demand paging rate, the size of the auxiliary storage manager's (ASM's) queue of pages to be moved, CPU utilization, real storage utilization, and average page delay time. These controls are of various degrees of usefulness, and all are at least somewhat flawed.

These deficiencies were remedied in SP 3.1.3 by removing the most inappropriate controls, and in MVS/ESA SP 4.2 with the introduction of the Working Set Manager. The principal objective of the Working Set Manager is to ensure optimum occupancy of central storage and to minimize wasted CPU cycles that served only to move pages in and out of central storage.

Most experts agree that the basic control of MPL in a system without expanded storage (or one with a workload segment excluded from expanded storage) should depend principally on the *page fault rate* (the rate of nonswap, non-VIO page-ins from auxiliary storage), with other parameters either nullified or set to values well outside a normal controlling range and thus responsive only to extreme problems. In a system with a homogeneous

paging configuration, with paging devices that all perform alike, page fault rate should be a good indicator of overall page-in delay.

A flaw in the original IBM definition of page fault rate was that it also included *reclaims*, stolen pages "rescued" from the page-out queue in response to subsequent page fault interrupts. Usually there are few reclaims in systems without expanded storage, but very high reclaim rates have been observed in systems with expanded storage. Reclaims were excluded from the page fault rate beginning with MVS/SP 3.1.3, and the attempt to perform reclaims was removed when block paging and other real storage management mechanisms were integrated in SP 4.2.

Each of the MPL adjustment parameters is discussed in the following paragraphs, followed by a summary table of default values and recommendations. They are presented in estimated order of importance.

RCCPTRT, page fault rate

Page fault rate remains the best available (to SRM) indicator of storage constraint and thus the most appropriate control variable for MPL. As of SP 3.1.3, reclaims were no longer included in RCCPTRT. In a default MVS system, RCCPTRT is set to (1000,1000), normally nullifying it as a control.[6]

There is no way that a definitive recommendation for RCCPTRT can be given *a priori.* The IBM default can thus be understood. The best we can do is to suggest values based on the ability of the paging configuration to resolve page faults in some acceptable time frame.

Based on modeled results, device limits are suggested (in the following table) for commonly used paging devices, based on target attainable device response times for several levels of underlying path busy (B_p) and device busy (B_d).[7]

6 Note that (1000,1000) is not a distinguished value causing a parameter to be disregarded. Sometimes the variables measured against those parameters do exceed 1000 and cause the MPL to decrease. For a parameter to be disregarded as completely as possible, it should be set to the "unfavorable" extreme value of its range, in most instances (32767,32767).

7 The correction for page-out activity is balanced by the efficiency gain from blocked paging. These numbers will do for a first guess.

Using these paging rates as guideline loading limits, one would choose the underlying device and path utilizations corresponding most closely to the actual paging configuration, read off the page-in rates, and multiply by the number of local page data sets, less one for each 100 logged-on TSO users in a system without swap data sets. The result will be a first approximation to the correct upper limit for RCCPTRT. Set the lower limit of RCCPTRT to 80 percent of that value.

Sustainable Paging Rates, Page-ins per Second								
Device	Response Time Target	B_p:	0%		10%		20%	
		B_d:	0%	10%	0%	10%	0%	10%
3380J	20 ms		31	25	30	24	29	22
3380K	22 ms		21	14	21	14	20	14
9345B12	15 ms		41	32	41	32	41	32
9345B22	15 ms		38	29	38	29	38	29
3390-1	15 ms		32	24	32	24	32	24
3390-2	15 ms		23	15	23	15	23	15
3390-3	20 ms		38	31	38	31	38	31
SSD[1] 3.0 MB/s	2.5 ms		147	95	90	47	73	32
SSD 4.5 MB/s	2 ms		273	129	125	70	105	52
[1] Solid state device, usually emulating an IBM 3380.								

Example: For the system described in the preceding section, assume that batch swapping uses the capacity of three of the eight 9345B12 paging devices. The suggested device limit for dedicated devices with 10 percent underlying path busy is 41 page-ins per second. Discounting the three aggregate devices used for swapping from the eight locals, the upper value chosen for RCCPTRT should be 5 X 41 or 205. The lower limit should be 80 percent of that, or 164. In the OPT, specify

```
RCCPTRT=(164,205)
```

After a reasonable period of measurement, RCCPTRT should be adjusted upward if unilateral swaps are triggered by a high page fault rate when the system is not suffering excessive paging delay. It should be made lower if paging delay is excessive and MPL reduction is not taking place to compensate.

RCCUICT, UIC

Controlling MPL based on UIC depends on the assumption that UIC is an indicator of storage constraint. In systems without expanded storage, this is rarely so; hence the extreme default of (2,4) for this parameter.[8] (UIC is the only MPL control for which low values indicate constraint.) Conceivably, low UIC can occur in a system with a stable and static workload and just enough real storage for it all to fit. In systems with more variable workloads and a high degree of real storage reuse, page stealing may commence before UICs get low enough to trigger MPL adjustment. Subsequent page faulting will cause MPL adjustment if RCCPTRT is set correctly. Again, the UIC control will not come into use.

In MVS/ESA with expanded storage, the picture changes. Pages don't move in if they don't move out. The integration of swapping with paging ensures that competing mechanisms don't churn superfluous paging traffic. Page stealing is innocuous if the page age is very high or if there is sufficient available expanded storage. Delay due to page-in from auxiliary storage is handled well by RCCPTRT as usual, but other sources of delay are possible. If the page movement rate to and from expanded storage is too high, CPU delay will rise significantly. Such excessive page movement is a symptom of central storage exhaustion, and low UIC is a moderately correlated indicator of that constraint.

Since MVS/ESA SP 4.2, with Working Set Manager, the value of RCCUICT should not be changed without thorough analysis—and a problem to solve.

RCCCPUT, CPU utilization

The idea of controlling the MPL based on CPU utilization seems to be consistent with the idea (a recurring theme in *Initialization and Tuning)* that SRM controls "access to resources." However, access to the CPU is controlled effectively by adjusting relative dispatching priorities. Moving address spaces in and out of the multiprogramming set by means of swapping according to CPU utilization has two effects:

8 The default for this parameter in MVS/SP 3.1.3 was set to (10,15). This was an interim response to the difficulty of detecting and dealing with central storage shortage. If you are converting from SP 3.1.3, make sure to change this parameter to the lower default. The Working Set Manager makes the SP 3.1.3 value an especially inappropriate choice.

- Constraint on central (real) storage varies according to system MPL, causing secondary SRM responses to deal with possible storage shortages. In MVS/ESA since SP 3.1.3, even this effect is muted because of the integration of real storage management functions. It's almost certain that a unilaterally swapped address space will linger in central storage, at least for a while.
- Swapped-out address spaces receive no service.

Denying service may be necessary under some special circumstances. Usually, however, the denial of service associated with MPL adjustment is too temporary to deal with disruptive address spaces—those causing contention with other, more important address spaces. Such an address space, if swappable, needs to be RESET via operator command to a special performance group denoting a domain with (0,0) constraints, to be *kept* swapped out until the potential for contention is removed.[9] (It may be just as effective to take an apparently contrary action: put the offender in a more favorable performance group with higher dispatching priority and a domain with generous constraints *if* it is a batch job with a predictably short time to completion.) In MVS beginning with Version 5, a **RESET QUIESCE** operator command provides a more elegant solution for dealing with disruptive address spaces.

Since the time of MVS/SP 2.1.7, the maximum value for RCCCPUT has been 128 rather than 101, as it had been in earlier MVS releases. With this limit, it is possible to "dial in" an acceptable range of CPU overcommitment. With a range of (103,107), for instance, SRM will allow MPL increase even if the CPU had no wait time and an average of fewer than three address spaces received no service during an SRM summarization (RM2) interval. MPL reduction will be forced if seven or more address spaces received no service during the interval.

Why should such CPU overcommitment be desirable? With adequate central storage, not needed for another purpose, it is harmless to leave inert address spaces in the multiprogramming set. Properly chosen dispatching priorities will control CPU access. Low-priority address spaces will be in a

9 Nonswappable "problem" address spaces can be even more disruptive. If contention is caused by mere ownership or allocation of resources rather than by usage, the address space may need to be shut down and restarted at some other time.

position to take advantage of any momentary drop in higher-priority CPU demand and will receive service without the delay of even the most efficient swap-in. In an MVS system with a mixed workload and a well-balanced hardware configuration, a modest level of planned CPU overcommitment should get the most out of the hardware.

With a more stable and homogeneous workload, restricting CPU overcommitment can ensure that I/O completion SRBs triggered by low-priority address spaces will be less likely to tie up the CPU when "important" address spaces need it. Such a consideration is valid only on uniprocessors.

In systems running under IBM's PR/SM LPAR, HDS's MLPF, Amdahl's MDF, or VM, the SRM does not see a correct picture of WAIT time unless the WAIT COMPLETE option (of PR/SM) or a corresponding parameter is selected. Since this option can be harmful to interactive response time, it is usually not chosen. Consequently, any MVS system running in a logically or virtually partitioned mode will be exposed to spurious MPL adjustment based on high CPU utilization, unless RCCCPUT is set to a range exceeding 100 for both the low and high values.

Summarizing the considerations for setting RCCCPUT:

- Basic recommendation: with adequate central storage and a TSO/batch workload, set RCCCPUT to (104,108) for systems with up to three processors, to (106,112) for a four-way, (107,114) for a five-way, and (108,116) for a six-way. Many systems run with "sponge" or "soaker" jobs,[10] one per CPU, to rule out "low utilization effects" and supply a direct measure of reserve CPU capacity. The base value of RCCCPUT should go no lower than 100 plus the number of CPUs, to allow for an undispatched cycle-eater in each processor. The upper value simply adds one to three additional standby address spaces per processor, since the sponge jobs are typically not swapped on account of being in domains with fixed constraints.

  The default values of RCCCPUT since MVS/ESA SP Version 5 are (128,128), reflecting the bias in recent MVS and

10 The idea may have been suggested first by Dr. H. W. "Barry" Merrill, who did much to popularize it. A sponge job is an unending simple CPU loop, running at the lowest dispatching priority. Its CPU utilization is what's left after all genuine workloads are satisfied. As many such jobs are needed as there are processors in the configuration.

OS/390 releases toward achieving maximum CPU utilization. The fair access dispatcher changes make it more likely that address spaces low on the queue will be dispatched occasionally. If your system has adequate central storage, leave the parameter at this default value.

- If in a partitioned environment (other than physically partitioned), make sure that RCCCPUT is at least (104,108) regardless of any other factors.
- Adjust both RCCCPUT values upward, one step at a time, if no significant CPU or storage delay is found in key workloads, and if address spaces are unilaterally swapped out, with RCCCPUT as a cause of MPL reduction.[11]
- Reduce the RCCCPUT range if this parameter rarely causes MPL reduction while workloads at relatively high dispatching priority experience significant CPU delay.

RCCFXTT, total fixed/allocated storage

The idea of controlling MPL based on the amount of real storage fixed or currently allocated for movement by the Auxiliary Storage Manager is appealing. Isn't a lack of available storage just what we're trying to prevent? A fallacy of this control is that the specification for RCCFXTT is not in frames but in percentage of frames. The default range of 66–72 percent, on a large LPAR with 800 megabytes of central storage, would result in MPL reduction with 57,344 frames neither fixed nor allocated for page-in or page-out. If this control causes spurious MPL reduction or prevents MPL increase, its values should be raised a little closer to 100 percent. However, check first to see if emergency page stealing is occurring frequently, or if the system-wide high UIC is chronically less than 10. If so, it would be more appropriate to reduce the RCCFXTT values than to increase them.

RCCFXET, fixed below 16 MB

This control, similar to RCCFXTT and similarly flawed, is designed to allow I/O with format-0 CCWs to proceed as it must, using only real storage below 16 megabytes. The need for this

11 Chapter 10 describes monitoring techniques to determine if such conditions exist.

control will eventually fade as I/O routines are switched over to Format-1 CCWs.

Summary of MPL adjustment parameters

The following table summarizes the preceding discussion of the SRM MPL adjustment parameters maintained in the OPT as of MVS Version 5.

Name	Default	Change to	Comment
RCCPTRT	1000,1000	Device-related	Most effective control
RCCUICT	2,4	Keep default	Adjust only if problem
RCCCPUT	128, 128	104,108 & up or leave at default	Allow full CPU loading; default OK since V5
RCCFXTT	66,72	Higher if problem	Avoid emergency steal
RCCFXET	82,88	Keep default	Verify necessity

B.1.7. Logical swap controls

Logical swapping was an opportunistic enhancement to MVS at a time when "large" real storage first became available. A very large available frame count was perceived as an indicator of unused excess capacity. Using those frames for logical swap improved average TSO response time and contributed to the marketing momentum supporting the proposition that MVS could make good use of practically any amount of real storage.

Given the marketing climate of the time, it is easy to see why the original controls in MVS/SE1 for logical swap were firmly biased to encourage logical swaps over physical. When those controls were made visible in the OPT in SE2 and in later releases, the now-discarded guideline of release-to-release compatibility dictated that the controls would not change.

Subsequent evolution has replaced the narrowly defined "logical swap" with "processor storage swap" in systems with expanded storage. The logical swap evaluation is made first, but if it fails, the trimmed working set of an address space may be retained in central storage on a pending push-out queue for expanded storage, to be moved out only when its central storage frames are needed as an alternative to a cycle of page stealing. In such a system, "turning off" logical swapping by setting the think time range to (0,0) may appear to be ineffec-

tive. It is effective in a strict and narrow sense; however, a pending processor storage swap to expanded storage looks just like a logical swap—hence the appearance that turning off logical swap is ignored.

Logical swapping is controlled by four OPT parameters. One (LSCTMTE) specifies the range of "think times" that the logical swap decision criterion may assume. The other three controls move the criterion up and down through the think-time range. The two controls on fixed and allocated storage are absolute: Think time will be reduced if either control (LSCTFTT or LSCTFET) is above its maximum, and it will not be increased if either storage control exceeds its minimum.

The UIC (LSCTUCT) control is permissive. Assuming no effect due to the storage controls, think time will be increased if the Unreferenced Interval Count is over its maximum. Think time will be decreased if the UIC is below its minimum. Conflict is resolved in favor of decrease. The UIC control serves another purpose: When the system UIC is less than half the lower limit, no new logical swaps will be allowed for the duration of the current RM2 interval.

Another use for the current think-time value is as an absolute control for making non-terminal-wait swaps logical. If the current think time is at least 5 seconds, such logical swaps are allowed.

Logical swapping defaults

The following box summarizes the parameters governing logical swapping and their default values.

Summary of Logical Swap Defaults (since SP 4.2)			
Name	**Default**	**Unit**	**Definition**
LSCTMTE	0,5	Seconds	Think time range
LSCTFET	76,82	Percent	Fixed frames below 16 MB
LSCTFTT	58,66	Percent	Fixed or ASM-allocated (all)
LSCTUCT	20,30	Seconds	Unreferenced Interval Count

Tuning with logical swapping

Because logical swapping is somewhat preferred by the default controls, those with storage-constrained systems may wish to reduce the preference. Two approaches may be taken: either reduce the think-time range or make the controls more disposed to reduce the think time. Both are usually done.

There is a caution that must be observed in this respect, as in all cases of tuning by means of adjusting the OPT controls. Any currently effective control will tend to place its controlling variable in its "happy"[12] range if the controlled variable is free to be adjusted. For example, suppose we set LSCTUCT to (40,50) and LSCTMTE to (1,5). If there is an abundant supply of logical swap candidates of varying think times, the system-wide high UIC will tend to settle somewhere between 40 and 50, with a think-time criterion somewhere between 1 and 5 seconds. If at the same time the UIC control of MPL (RCCUICT) is set to its default of (2,4), the free adjustment of MPL will tend to keep UIC between 2 and 4. The UIC cannot be both between 2 and 4 and between 40 and 50. In this case, it will probably rise to the higher range, and UIC will be ineffective as an MPL control (as it usually is). If storage activity increases, UIC will fall to the lower range, leaving think time pinned at its lower limit but serving then as an effective MPL control.

This interaction between free and controlled variables, and the switching of control from one area to another, is not necessarily inappropriate; it just needs to be understood. If the ranges had been reversed, SRM might have caused oscillation or other signs of instability as it tried to reconcile the conflicting demand to reduce MPL while increasing logical swapping, both based on UIC.

Typical logical swap adjustments

Following are a few examples of adjusting the logical swap controls from their default values.

In systems without expanded storage and with some degree of real storage constraint, it is common to alter the logical swap

12 OPT controls were often called "happy values" in the past. "Happy" referred to the condition when the controlling variable was between the lower and upper limits, resulting in no change to the controlled variable. A more apocryphal origin goes back to the days when there were essentially no OPT parameters and SRM controls were adjusted by ZAPping values in MVS control blocks. To alter such a parameter successfully (without crashing the system) and effectively (producing the desired result) was a happy occasion.

controls to reduce the amount of logical swapping under normal conditions, and to cut it off before delay becomes unacceptable. The following are typical and usually produce the desired results:

```
LSCTMTE=(1,5)
LSCTUCT=(30,40)
```

In systems with expanded storage, logical swap is absorbed into processor storage swap. The logical swap controls are irrelevant if left at their default values, and can cause great harm if altered. In spite of past advice or practice to the contrary, the think-time range should not be increased above its default values.

B.1.8. CPU mean-time-to-wait adjustment

One OPT parameter originally used to control CPU load balancing (dropped with the other load balancers as of MVS/ESA SP 4.1) has another use—to set the interval of mean-time-to-wait associated with each dispatching priority step in the MTTW algorithm. The parameter is CCCSIGUR; it has a default value of 45, in units of SRM milliseconds. If several dissimilar address spaces are in MTTW group *x* and their actual dispatching priorities, as seen with a performance monitor, are not spread out in the range *x*0 through *x*9, an adjustment to CCCSIGUR could ensure that MTTW works properly.

If the priorities are clumped at *x*9, it appears that all address spaces have short MTTW. Thus CCCSIGUR is too large and should be decreased. If all address spaces have the lowest dispatching priority in the group, CCCSIGUR is too small and should be increased. Start by doubling or halving the value, then make smaller adjustments until the desired result is achieved.

B.1.9. Selective I/O enablement control

In systems with multiple CPUs, MVS/XA and MVS/ESA start normal processing after IPL with only one CPU enabled to handle I/O interrupts. An additional CPU is enabled when the percentage of I/O interrupts handled by the TEST PENDING INTERRUPT (TPI) instruction exceeds the upper threshold of the CPENABLE OPT parameter. A CPU will be disabled for I/O interrupts when the TPI percentage falls below the lower CPENABLE threshold. The CPENABLE defaults are (10,30).

If CPENABLE is to be changed from its default values, the following effects must be considered:

- If the CPENABLE range is increased, the CPU(s) not handling I/O interrupts will run more efficiently by avoiding status-switching overhead. There will be a small increase in I/O completion time owing to CPU queuing.
- Decreasing the CPENABLE range will minimize CPU queuing in handling I/O completions at the expense of increasing status-switching CPU overhead on all enabled CPUs.
- In a logically partitioned system, a decreased CPENABLE range might cause extra switching of processors between partitions or domains, with whatever overhead the hardware and microcode impose, as physical processors are continually reassigned among logical processors. However, if the partition or domain is receiving only a small fraction of the processor complex's service, setting CPENABLE to lower values will minimize I/O elongation.

If the most important workload on the system is heavily dominated by I/O, reducing the CPENABLE range might improve I/O response times to a slight extent.

B.1.10. Miscellaneous controls

Certain additional and largely unrelated controls are in the OPT. We consider them briefly:

- **CNTCLIST**, a **YES/NO** variable with a default of **NO**. If **YES** is specified, the individual commands in a TSO CLIST are counted as individual SRM transactions. The default of NO counts the entire CLIST as a single transaction. Specifying **CNTCLIST=YES** tends to increase the number of reported transactions and reduce the reported "response time."

  One application of **CNTCLIST=YES** is to ensure preferential treatment of large CLISTs that constitute end-user applications or that are invoked by LOGON procedures.

- **DVIO**, a **YES/NO** variable with a default of **YES**. The default allows "directed VIO." VIO page-outs will go only to page data sets not designated as NONVIO in IEASYSxx or in a

PAGEADD command. Specifying **DVIO=NO** defeats directed VIO even if the NONVIO designation is used for one or more page data sets.

Directed VIO can be beneficial in a mixed paging configuration if VIO is to be kept off relatively small high-speed paging devices. It should not be used with homogeneous paging configurations.

When sufficient expanded storage is in the system, VIO will normally go to expanded, and DVIO becomes irrelevant.

- **ERV** (ENQueue residence value) specifies a number of CPU (TCB) service units an address space must accumulate before it may be considered for a unilateral or exchange swap-out if it holds an ENQ on a resource needed by another address space, or an exclusive ENQ leading to a device RESERVE. The default **ERV** is 500 CPU service units. During the **ERV** interval, the address space or enclave (as of SP 5.2) runs at the "privileged" dispatching priority.

- **MCCFXEPR** and **MCCFXTPR**, percentage variables with defaults of 92 and 80, respectively. The first applies to fixed frames in central storage with real addresses below 16 megabytes; the second to fixed pages, or those allocated for page-in or page-out, in all of central storage. If either threshold of real storage allocation is exceeded, SRM will signal a pageable storage shortage and begin a series of actions described in grisly detail in *Initialization and Tuning Guide*. If the pageable storage shortage condition occurs with any regularity (*i.e.*, more than once a week), the thresholds might be raised slightly and the cause investigated. If a pathological condition is responsible, raising the thresholds will have little effect.

- **RMPTTOM** specifies a scale factor for SRM invocation. The default and minimum is 1000, signifying a normal 1000 SRM milliseconds per SRM second. RMPTTOM might be increased in an MVS test system running under VM, to reduce overhead somewhat. It should not be increased in a production MVS system running in a preferred, V=R, or V=F virtual machine.

  It might be desirable to increase RMPTTOM in an MVS system with a stable nonswappable workload, such as an

all-CICS system. A minor overhead component will be reduced. It is *imperative* if such a choice is made to make sure that there is no swappable work of any importance on the system.

B.1.11. OPT questions

1. Examine the current OPT in your MVS system. Fill in the default specifications if they're not there. If there is no OPT, fill in all of the defaults. Identify the parts that control access to expanded storage, the multiprogramming level, and logical swap. Discard the load balancers except for CCCSIGUR.
2. Is RCCPTRT the most effective control of MPL? Does CPU utilization restrict MPL? Is the UIC range (RCCUICT) too high? Review the discusssions in this chapter and make appropriate adjustments.
3. Examine the RMF Swap Placement Report to determine the effect of logical swap management as specified in your OPT. Can you distinguish between "true" logical swaps and processor storage swaps that didn't get pushed out? What changes would you make so that the system will perform according to your expectations for logical swapping?
4. There should be two UIC ranges in your OPT. Determine the range of UIC values for each of the following:
   - MPL decrease
   - no MPL change
   - MPL increase
   - logical swap think time increase
   - no change in logical swap think time
   - logical swap think time decrease
   - no logical swap (cut-off)

   Are there any inconsistencies or unexpected results? What adjustments would correct them?
5. What is your installation's critical migration age? If you don't have expanded storage, assume it is 700 seconds. Determine how many different classes of work you have for the purpose of determining expanded storage eligibility.

Now set up the criteria table entries for your installation's workloads.

B.2. Installation Performance Specification (IPS)

An IPS tells SRM how to treat each address space in the system, and to some extent how to treat the common area as well. Within the overall bounds and parameters established by the OPT, IPS parameters specify how MPL is to be distributed, the ranking of address spaces in competing for CPU cycles, the degree to which each address space and the common area is subject to page stealing, and the eligibility of each address space's pages to be sent to expanded storage.

The IPS contains several kinds of information in a particular order. Global definitions precede domain and performance group definitions.

IPS concepts were discussed at length in Chapters 6 and Appendix A. Here we'll concentrate on syntax, defaults, and recommended values. Note that there are two kinds of defaults in the IPS:

- basic default—the value assigned if the parameter is omitted in the IPS
- supplied default—the value specified in the default IPS supplied with current MVS releases

The basic defaults are unlikely to change except at version changes; the supplied defaults have evolved to some extent over time.

B.2.1. Global definitions

Global definitions include service definition coefficients, which must come first, followed by specifications for privileged dispatching priority, time unit divisor, time-slice pattern, I/O queuing option, common-area storage isolation, and I/O service unit option.

All of these parameters are optional. Defaults exist for all but the time-slice pattern.

Service Definition Coefficients (SDCs)

- **CPU** may range from 0 to 99.9. The "hard-wired" (basic) default is 1.0 if the coefficient is not specified, but a default of 10.0 is used in the supplied IEAIPS00 in an MVS/ESA system as distributed.

  Recommendation: Use the supplied default value of 10.0.

- **SRB** may range from 0 to 99.9. The basic default is 0.0 (because SRB service was not captured 'way back in MVS/SE1) if the coefficient is not specified, but a default of 10.0 is used in the supplied IEAIPS00 in an MVS/ESA SP 4.2 system as distributed. The value of the basic default makes it important that this parameter be specified in every IPS.

  For various reasons, those at some installations set the SRB coefficient to a different value than that of the CPU coefficient. This might be understandable if the SRB value were made greater than the CPU value, since SRBs have higher effective dispatching priority than TCBs, and thus a greater impact on workloads. Unfortunately, a faulty assessment of "importance" may be used, resulting in an SRB value lower than the CPU value.

  Recommendation: Use the supplied default value of 10.0. Maintain equal values for the CPU and SRB coefficients.

- **MSO** may range from 0.0000 to 99.9999. The basic default is 1.0 if the coefficient is not specified, but a default of 3.0 is used in the supplied IEAIPS00 in current systems as distributed.

  Recommendation: In most cases, simply set MSO to 0.0 and be done with it. Even though SMF Type 30 records now report raw page-seconds, your reporting and analysis methodology may still make some use of the MSO number. If there is such a need to preserve the MSO service number, flawed as it is, set MSO to 0.01 or less.

- **IOC** may range from 0 to 99.9. The basic default is 1.0 if the coefficient is not specified, but a default of 5.0 is used in the supplied IEAIPS00 in an MVS/ESA system as distributed.

  Recommendation: Use the supplied default value of 5.0. It is unlikely that any alteration of IOC will be necessary.

Other Global IPS Parameters

- **PVLDP** is the dispatching priority for privileged address spaces, including initiators at job and step transitions, TSO LOGONs, and other address spaces denoted as "privileged" in the *Program Properties Table* (*PPT*), as well as address spaces and enclaves in their ERV protected interval. Its basic default is **PVLDP=M0** (the lowest SRM-owned dispatching priority), but it is specified in the default IPS as **PVLDP=F54**. As with any other dispatching priority, the value is meaningful only in context with other priorities. In the supplied default IPS, F54 ranks higher in dispatching priority than all workloads including first-period TSO; initiators with this priority might cause TSO delays at job step transitions.

  Recommendation: Make PVLDP one step lower in priority than the lowest-priority response-critical workload. For instance, if TSO first period is at F72 and production CICS address spaces are at F71 and F70, set PVLDP to F7.

- **TUNIT** need be specified only if time slicing is to be used, and then only if the SRM second on the current CPU is not short enough to execute the time-slice pattern without causing perceptible delays in some workload. TUNIT ranges from 1 to 10; the SRM second is divided by TUNIT to become the SRM time unit.

  Recommendation: Avoid using TUNIT if at all possible. There is an overhead impact. If you must use it, make TUNIT the smallest (least impacting) value needed to smooth out priority-associated response time fluctuations if they are noticed.

- **TSPTRN** is required if time slicing is to be used. One such circumstance is the need to divide CPU preference among different subsystems or user subsets to satisfy the terms of a service level agreement.

  Assume, for example, that TSO users in performance period 1 of performance group 12 are to be preferred 30 percent of the time and those in period 1 of performance group 14 are to be preferred 50 percent of the time, regardless of the relative activity in the two performance groups. The time-slice pattern should contain ten slices, five going to

time-slice group 1 (PG 14) and three to time-slice group 2 (PG 12). The remaining two are unassigned.

The simple assignment of

```
TSPTRN=(1,1,1,1,1,2,2,2,*,*)
```

would cause each user set to wait for five SRM seconds before gaining preference, and other address spaces would need to wait for eight slices before no address space would be at its time-slice priority. A simple rearrangement minimizes such delay and makes a subdivided (*via* TUNIT) SRM second unnecessary:

```
TSPTRN=(1,2,1,*,1,2,1,*,1,2)
```

Recommendation: Avoid time slicing if at all possible. If it is found to be necessary, try to minimize complexity by avoiding the need for the TUNIT parameter, as shown above.

- **IOQ** has two possible values, *FIFO* (the base default) or *PRTY*. **IOQ=PRTY** is specified in the IEAIPS00 shipped with MVS/ESA.

  Recommendation: **IOQ=PRTY** is usually specified and can have a beneficial effect if there is a great deal of low-priority, I/O-intensive batch running along with other workloads and causing I/O contention with those other, more time-critical workloads. Since there seems little possible harm in making this choice, with some possibility of benefit, it is recommended.

- **CWSS** is used to specify storage isolation for the common area—SQA, PLPA, and CSA, as well as their "extended" counterparts above 16 megabytes. The base default is (0,*), or no storage isolation. The lower value specifies a minimum number of frames never subject to ordinary page stealing; the upper sets a number of frames above which the common area is subject to preferred page stealing.

  If CPGRT is not specified, the protected frame count remains at the lower value. With CPGRT, the number of protected frames (the *target*) is adjusted between the minimum and maximum according to the rate of page faults resolved from auxiliary storage.

  Recommendation: If you use CWSS, *always* set the maximum to "*," denoting the largest possible value for the pa-

rameter and preventing preferred stealing from the common area. Set the minimum to the observed minimum sum of frame counts in the various common-area components at a relatively un-busy time of day. With expanded storage, the common-area paging rate from auxiliary storage is likely to be very low; CPGRT would have little effect and should not be used.

Without expanded storage, follow the recommendations of the next paragraph to set CPGRT.

- **CPGRT** sets a range of acceptable page-in rates for the common area. If the rate from auxiliary storage exceeds the maximum, the target is increased. When the page-in rate falls below the minimum, the target will be reduced, but not below the minimum CWSS value.

  Recommendation: Without expanded storage, set CPGRT to a range that will lead to an acceptably low level of common page-in delay. Use the techniques described in Chapter 10 to make this determination. When starting with storage isolation, set CPGRT to somewhat high values, such as (10,15), and observe the target value of CWSS. If the target never rises above the minimum, and unacceptable common page-in delay continues, reduce the range, maintaining the maximum at 1.5 times the minimum, until the CWSS target rises when storage constraint develops.

  CPGRT may be omitted in systems with expanded storage unless the amount of expanded storage is very inadequate.

- **IOSRVC** may be specified as COUNT (the default) or TIME. The considerations for choosing are discussed in Chapter 6, in the section headed "I/O service units" on page 144.

B.2.2. Domains

Domains are defined with up to four parameters, only the first of which, denoting the domain number, is required. The other three parameters specify the MPL constraints on the domain, the means of determining the domain's contention index, and the index of expanded storage criteria table entries to be used for address spaces in the domain.

A domain defined without constraints has a default range of (1,999). If no contention index algorithm is specified, a DSRV

range of (0,999999999) is assumed. The default constraints allow wide SRM control of MPLs, in general a good strategy. A default lower constraint of 0 rather than 1 might be better; the extra increment of control is crucial in controlling storage-intensive workloads.

Two default domains are predefined:

```
DMN=0,CNSTR=(999,999),DSRV=(0,999999999)
DMN=1,CNSTR=(1,999),DSRV=(0,999999999)
```

Domain 0, for privileged address spaces, cannot be altered, nor can it be explicitly used for a performance group. Domain 1 is an ordinary domain; it may and should be redefined to suit the installation's needs.

Recommendation: Review the discussion of domains in Appendix A, as well as the material on the expanded storage criteria table earlier in this appendix.

- Create the minimum number of domains needed to manage each distinct workload component, considering both MPL and expanded storage eligibility ranking.
- Omit constraints on domains serving nonswappables only.
- Ensure that each swappable domain's MPL is free to vary over as wide a range as possible.
- Set nondefault minMPLs only in domains supporting response-critical swappable address spaces, such as first- or second-period TSO.
- Use the default range of (1,999) in most cases, with zero minMPLs and restricted maximums reserved for very storage-intensive workloads.

B.2.3. Performance groups

All that has gone before in the IPS and in our consideration of its parameters is mere prelude to its real business—the definition of performance groups for the purpose of managing groups of address spaces In the performance management model we wish to encourage, each address space is assigned to a performance group through the ICS, described at the end of this appendix. The ICS is the bridge between the outside world of job names, job classes, and user-ids, and the SRM's view of address spaces, each assigned a performance group number.

The definition of a performance group is simplicity itself:

PGN=*number*

defining a number in the range 1–999 as an acceptable performance group number to be a target of assignment in an ICS, a PERFORM specification in JCL or a TSO LOGON command, or an operator RESET command.

Each IPS *must* include definitions of performance groups 1 and 2; all others are optional.

If only the minimal definition is provided, default assignments are made for domain (domain 1), dispatching priority (M0), I/O priority (M0), storage isolation (none), time slicing (none), and RTO (none). If the default parameters are not acceptable (as of course they will not be for most performance groups), additional specifications are needed.

Performance Periods

Alterations to the default address space management parameters are added to the basic performance group definition as separate groups of value assignments enclosed in parentheses. The first of these *performance period* definitions is set off from the performance group name by a comma. Additional period definitions are delimited by enclosing parentheses with no intervening comma. As many as eight periods are allowed in a performance group.

Each SRM transaction starts execution in the first period of its performance group. As service or time accumulates, SRM advances the transaction to the next period after a sufficient accumulation is noted. As of MVS/ESA SP Version 5, nonswappables are also moved through periods. The following sections describe the set of parameters that may be defined for each performance period.

Performance Period Parameters

- **DP=** is the way to specify the dispatching priority of each address space in a performance period. The details of specifying DP are covered in Appendix A. If the DP parameter is omitted in a performance period definition, a priority of M0 is assigned.
- **DMN=** designates the domain in which address spaces in this performance period will be managed. If the parameter is

omitted, domain 1 is assigned by default unless there is a prior period in the performance group. In that case, the domain of the prior period is selected. There is no need to specify a domain for a performance group serving only nonswappable address spaces, unless the workload is to have its own expanded storage criteria. Whatever the definition of domain 1 (assigned by default if no DMN= appears), it has no effect on a nonswappable address space. When domain 1 contains swappable address spaces as well as nonswappables, its constraints should take into account both categories of address spaces.

- **DUR=** specifies the minimum duration of the performance period. DUR *must* be specified in each period of a multiperiod performance group except the last period. Its value (specified without commas) may range from 0 to 999,999,999 or 999,999K. The unit of the value is SRM service units, unless UNT=R has been specified in the period definition as well. In that event, the unit is in real-time seconds and is limited to a maximum value of 1,000,000. No error is indicated if the limit is exceeded.

  The actual duration may exceed the number of service units specified; the SRM determines when to switch periods through sampling of address spaces and performs the switch the first time it detects that the duration of the current performance period has been exceeded.

- **UNT=** can alter the unit of DUR from service units (**UNT=S**, the default) to seconds in real time (**UNT=R**). **UNT=R** can be used for such clever tricks as ensuring that a transaction that exceeds a certain number of service units is subjected to a punitive delay:

```
DMN=86,CNSTR=(0,0)
 ...     ...
PGN=2,(...,DUR=400,...)
  (...,DUR=4000,...)
  (DMN=86,DP=MO,...,UNT=R,DUR=60,...)
  (...)
```

Transactions ending in period 1 or period 2 will end normally; those going over 4400 service units will be subjected to a minute in the "penalty box," swapped out. After the

1-minute delay, a transaction is allowed to proceed. At some MVS installations, this technique is used to "discourage" TSO users from executing long-running transactions (such as program compiles) from the terminal.

Imposing such a procrustean judgment on the behavior of online users is most emphatically *not* recommended. Many TSO transactions other than compiles can often consume a large number of service units.[13] A well-meaning attempt to discourage apparently wasteful behavior can result in "innocent" users being subjected to unacceptably poor response time.

Another issue arises here: What might happen if the Data Center Manager said to the Manager of Financial Applications Development, "I think all of *your* programmers should submit all of their compiles as batch jobs. It makes *my* system management job easier. If they don't, I'll delay their long TSO transactions by a minute each." The M.F.A.D. might reflect on relative priorities, and their next meeting might take place in someone else's office—say, that of the Vice President of Corporate Services.

The action of imposing an arbitrary delay on inconveniently lengthy transactions is entirely equivalent to such arrogant words. It is all too easy to conceal an unacceptable policy position if it is embodied within the mystery of an IPS.

A less controversial application of UNT=R is a variation on the frequently used swap-out performance group. If the problem for which the offending job is to be swapped out is temporary, it reduces the burden on the operator to let the swapped-out period end automatically. A performance group like the following might be used:

```
DMN=13,CNSTR=(0,0)
...
PGN=72,(DMN=13,DP=MO,DUR=900,UNT=R)
       (DMN=1,DP=MO)
```

13 A given transaction may vary in size, especially if the default MSO SDC is retained. When storage demand is low, the count of MSO service units will be higher than when demand is high. The punishment is more likely when the behavior is less likely to do harm.

In this case, the penalty after a RESET to PGN 72 ends after 15 minutes, and the job is allowed to run as domain 1 permits.[14]

- **IOP=** is in the same form as the DP parameter. Normally (by default), the I/O priority of an address space is equal to its dispatching priority. A different I/O priority may be selected with the IOP parameter. Address spaces in a mean-time-to-wait group have unchanging I/O priorities. IOP=F*n* is not allowed.

  Recommendation: Always specify **IOQ=PRTY** to gain the benefits of priority I/O queuing. Accept the IOP default unless and until analysis of workload problems indicates that a key workload is suffering excessive I/O queueing delay that cannot be relieved by normal responses such as removing contending I/O from a device. As a last resort, a higher I/O priority may help. On the other hand, if a workload causing I/O interference with another is safely meeting its service target, giving it a lower I/O priority might help both workloads to succeed.

  The possibility of improvement is small, and more direct methods are usually more effective.

- **RTO=** may be specified only in the first period and applies only to TSO performance groups. Its purpose is to ensure uniformity of TSO response time for short-running transactions. If the average response time in the first period is faster than desired, an RTO may be specified, denoting a somewhat higher target. The greater the gap between the native response time and the specified RTO, the more uniform will be the resulting response time—in theory.

  Unfortunately, it's not that easy. The idea of RTO is flawed in the first place if current evidence linking minimum (subsecond) response time to enhanced TSO-user productivity is to be believed and applied. RTO was dropped as of MVS/ESA SP Version 5.

  Recommendation: The only plausible reason for using RTO in a production TSO system is if a system's workload is expected to increase substantially in the future and then to

14 Thanks to Cheryl Watson of Watson & Walker for suggesting this useful idea.

impose enough constraint to result in increased TSO response time. In that case, specify an RTO close to the expected ultimate trivial TSO response time, so that response time expectations may be properly conditioned.

In a system with a clearly subordinate TSO workload, an RTO of 1.5 to 2.0 seconds might discourage the TSO users enough to make them abandon TSO in favor of another subsystem or to use personal computers instead. Do this if you must, but don't be surprised if the workload leaves the system.

- **PWSS=** is used to specify storage isolation for each address space in a performance period. Along with dispatching priority and the ESCRTABX parameter on the domain definition, it is one of the few IPS controls applicable to nonswappable address spaces. PWSS is specified as two values defining a range for a target protected working set in frames. Both values are in the range 0 to 32,767 in MVS/370 and MVS/XA, and 0 to 2,147,483,647 in MVS/ESA. The maximum value in either case may be denoted by "*."

  Storage isolation may be static or dynamic. If neither PPGRTR nor PPGRT is specified along with PWSS, the target is always the minimum value specified. If either of the paging-rate controls is added, the target varies between the minimum and the maximum according to the page-in rate of the address space.

  The target applies to the sum of frames in central and expanded storage. An address space at or below its target working set is protected from ordinary page stealing. If its combined frame count exceeds the target, stealing is done down to the target at the time the address space is reached in the normal ASID-ordered scan of address spaces. If an address space's frame count exceeds the maximum, it is targeted for preferred page stealing in a preliminary address-space scan of the page stealing algorithm. All frames above the maximum are vulnerable, up to the currently required number of frames.

  One further aspect of storage isolation is noteworthy: when a storage-isolated address space is swapped out, swap trim does not go below the minimum PWSS value. In other words,

the minimum target working set is preserved across swap-in and swap-out. This property defines the *swap group* size and is of crucial importance in managing TSO paging delay, particularly in systems without expanded storage and with some degree of storage constraint.

Most page faults in TSO transactions (60 percent or more) are re-reference page faults, bringing in pages formerly used in the session. If storage isolation is specified for response-critical TSO performance periods (usually first and second) at a value somewhat above the mean frame count for active transactions, re-reference page faults will be reduced drastically, at the expense of a few extra frames being tacked on to the last swap set.

For example, if an all-locals paging configuration on 3380s is used, the effective swap set size is 30 slots. If the mean frame count is 103 (by "eyeball averaging" on a performance monitor's display), four swap sets will be needed for the average swap-out. Specifying PWSS=(120,*) does not increase swap I/O, except to increase the average number of pages transferred in the last swap set, at a cost of 1.34 milliseconds per page. Page faults might be reduced by 3 to 6 per transaction, at about 50 milliseconds per page fault. The need for this technique has declined with the availability of block paging.

Recommendations: Storage isolation has numerous benefits and almost as many pitfalls. Several recommendations are in order:

- For nonswappable subsystem Loved Ones such as CICS, specify storage isolation to control paging delay in systems without expanded storage or to guard against migration in systems with expanded storage. A minimum PWSS may be used to ensure minimal delay following a period of inactivity. A limiting maximum PWSS *should not be used* for a preferred workload, and dynamic adjustment of the target according to page-in rate should be used. For a CICS address space,

  ```
  PGN=14,(PWSS=(200,*),PPGRTR=(1,3),...
  ```

  is a typical specification.

- For TSO performance groups that will not be swapped to expanded storage, use PWSS to define the swap group size. A paging-rate control is not appropriate, nor is a restrictive maximum. PWSS=(144,*) is a typical starting point. If frame counts frequently exceed the target (or if unacceptably high levels of page-in delay are observed), raise the PWSS minimum to the next multiple of the swap set size.
- One of the many performance management aphorisms attributed to those in the IBM Washington Systems Center is, "Know who your loved ones are—and always have someone else to kick around!" For address spaces to be fitted with *KICK ME* signs, a special form of storage isolation ensures that their frames are among the *first* ones stolen, thus protecting respectable workloads from such treatment.

  For nonswappable and distinctly nonpreferred address spaces, specify *negative* storage isolation by specifying PWSS with a zero lower limit and an upper limit less than the normal working set of the address spaces in the performance period. For batch jobs normally needing about 300 frames, PWSS=(0,160) will make about 140 frames of each address space in the performance period available whenever page stealing is needed. Page-in delay in the system will move toward these address spaces and away from other workloads.

  The same treatment is applied automatically to swappable address spaces by the Working Set Manager in MVS/ESA SP 4.2 and later systems. The Working Set Manager will not monitor or manage storage-isolated address spaces, so negative storage isolation should *not* be specified for swappable address spaces.

- **PPGRT[R]=** is the other part of specifying storage isolation. It is used to adjust the target working set according to page-in rate. When the page-in rate exceeds the upper limit of PPGRT or PPGRTR, the target is raised. A page-in rate less than the lower limit causes the target to be reduced. With a page-in rate between the limits, the target is not adjusted.

PPGRT was the first form of this parameter available; it is still accepted in MVS/XA and MVS/ESA. PPGRTR was added early in MVS/XA. The paging rate specified in PPGRTR has a uniform meaning for all kinds of address space: page-ins per resident second of real time. Such paging rates are easy to read directly from most performance monitors.

Recommendation: Use PPGRTR if page-in rate control is to be used. Do not use PPGRT. For example, if a CICS address space can tolerate four page-ins per second, the upper limit for PPGRTR should be 4. The lower limit is usually 50 to 80 percent of the upper limit. For our CICS address space, use

```
PPGRTR=(3,4)
```

As with all controls of this nature, PPGRTR should be adjusted so that in normal operation the target protected working set is not stuck at either extreme of the range.

- **TSDP=** is the alternative, higher dispatching priority when time slicing is used. The form of the time-slice priority must be the same as the form of the base priority: if **DP=F52** is specified, **TSDP=F71** or **TSDP=F9** is acceptable, but **TSDP=M7** is not.
- **TSGRP=** designates the time-slice group for the performance period.

B.3. Installation Control Specification (ICS)

The ICS is not strictly an SRM control; it serves simply to assign performance groups to address spaces. Since there is no ICS by default,[15] it is essential to create and use an ICS so that entry to each performance group is controlled as desired. The ICS is organized in a two-level hierarchy. A SUBSYS statement introduces each section for batch (normally the primary subsystem), for TSO, for APPC/MVS transactions, or for started tasks, as well as for secondary subsystems that create address spaces. A default PGN (performance group, called in this context a *control* performance group) is usually specified on the SUBSYS statement. Also allowed are definitions for a default report perfor-

15 There is, sort of. Without an ICS, TSO sessions are assigned by default to performance group 2 and other (nonprivileged) address spaces to performance group 1. Privileged address spaces go to the mysterious performance group 0.

mance group (RPGN) and for one or more optional control performance groups (OPGN) applying to each address space in the performance group.

Following each SUBSYS statement may appear additional statements defining exceptions to the SUBSYS defaults, defined on the basis of user-id, job class, address space name, or accounting data. For each such subset of address spaces, PGN, RPGN, and OPGN may be specified in any combination. In current ESA releases of MVS, name specifications may include a "wildcard" or "mask" character, specified as a single character in a MASK statement that must come first in the ICS if it is used.

The ICS is well documented in *Initialization and Tuning*, so its parameters will not be discussed in detail here. Note, however, that the order for assigning performance groups to transactions changed in MVS Version 4. If converting from a prior release to Version 4 or later, evaluate carefully the hierarchy diagram in the *Initialization and Tuning Guide* under the heading "Assignment of Control Performance Groups."

Recommendations:

- Above all, make sure that there *is* an ICS. Without one, a carefully constructed scheme for controlling dispatching priority and other execution attributes is at the mercy of every batch job and the distributed JCL for every started task, unless well-designed exits screen JCL and enforce correct performance group assignments.
- The OPGN parameter is often ignored. Its use suggests an uncertainty about which address spaces should be in which performance group. If the OPGN is perceived to be more favorable than the base PGN, it will be selected with little consideration. An OPGN less favorable than the PGN is unlikely to be used at all. To the extent that a choice is allowed, it should be one with clearly defined benefits to each possible selection under conditions known only to the person responsible for the address space.

  For instance, if there are at least two different TSO performance groups, PGN might denote a performance group with steeply declining MPL and dispatching priority in later periods but a high MPL in the first and second periods. OPGN would designate a performance group with more gradual decline in favor and more generous storage

isolation (in pre-SP 4.2 systems), but a lower MPL for first-period transactions. Users planning to do only editing of programs or text might receive better service in the default performance group, while those intending to use a fourth-generation or query language, or to initiate frequent online compiles or text processing runs with BookMaster, might choose to use the OPGN. In each case, the wrong choice means poorer service.

- Use the ICS to assign each started task and each APPC/MVS transaction to an appropriate performance group. The default assignment should have relatively low dispatching priority, similar to that of ordinary batch. Be aware of installation naming conventions and conform the STC and ASCH TRXNAME entries to them, using the MASK capability if available.

  Individual control of started tasks' performance groups is even more important in an MVS/ESA SP 4.2 or later system with expanded storage. The individual assignment of tasks to distinct performance groups is a necessary step on the way to associating them with domains, in which the ESCRTABX parameter can direct each to its customized set of expanded storage criteria table entries.

  Some started tasks, even if assigned to particular performance groups, will not pick up all of their characteristics. The GRS address space, for instance, operates at a predetermined dispatching priority. The benefit of assigning GRS to a performance group is that its default storage isolation can be changed. GRS prior to Version 5 appeared to have a default storage isolation something like PPGRT=(0,0); its target working set was increased by any page fault and was (almost) never decreased. The special storage isolation for GRS was removed in Version 5.

B.4. Other SRM Controls

In editions of *Initialization and Tuning* predating MVS/ESA SP 4.2, there is a page at the end of Chapter 5 labeled "SRM Constants," describing values for some thresholds not found in the OPT.[16] In the rare event of a problem in such an area, it might

16 This page seemed to have dropped out of sight in the MVS/ESA Version 4 edition.

be appropriate to use the SUPERZAP service program to alter these values. Most of the frequently altered SRM controls have been added to the OPT over time.

One necessary parameter still buried without externals is the two-level threshold against which the number of in-use auxiliary storage slots is measured. The less disruptive lower threshold, MCCASMT1,[17] has a default value of 70 percent. When more slots than this percentage of all slots are allocated, the SRM bars new work (such as TSO LOGONs) from entering the system. At the second threshold, MCCASMT2, 85 percent by default, address spaces acquiring auxiliary storage are swapped out. The system may eventually become paralyzed.

In today's systems with very large page data sets, especially when expanded storage carries the bulk of the paging burden, these thresholds are far too restrictive and, if exceeded, can cause periods of great disruption. Adjusting these thresholds so that a reserve of perhaps 20,000 slots is maintained should be safe in most systems. There is a severe penalty for being wrong in this case, spelled I-P-L. Approach this adjustment with great caution.

A second area controlled from within the MCT is the detected wait threshold, at which an address space in a protracted non-LONG WAIT is made eligible for swap-out. The controlling field is MCCMS6L, which has a default value (in milliseconds) of 2000. Some special workloads may benefit by having this threshold changed either upward or downward. Increasing the value would tend to prevent swaps but increase any central storage constraint. Decreasing it would make swaps more likely but keep idle address spaces out of central storage if the frames are needed. If reducing the threshold, check the table in the *Initialization and Tuning Guide* of SRM seconds per real-time second. The value cannot be reduced below 8 SRM seconds.

B.5. Other Controls

The *Initialization and Tuning Reference* describes many members of SYS1.PARMLIB other than those controlling the SRM. Many of them can have a profound effect on MVS performance. It is beyond our intended scope to make recommendations for each of these parameter sets. We leave these as an exercise for the reader, with the

17 This field, as well as MCCASMT2 and MCCMS6L, may be found by name in the MCT control block, as mapped by the *IRAMCT* macro.

plea that an analysis of potential performance impact be added to the planning cycle.

This principle must be extended to subsystem and application planning as well. Choices such as buffering options, residency and page-fixing options, the use of STEPLIBs, and many others need to be put in context with priority, resource availability, cost, and service level agreements. While it is true that most performance problems are responsive to analysis of workload delays and tuning to minimize delays, planning and impact evaluation can shorten the time needed for workloads to stabilize with acceptable performance.

B.6. Comprehensive OPT, ICS, and IPS Example

Let us return to the system introduced in a partial OPT example earlier in this appendix. We'll also reiterate the principles of "the well-ordered CPU" introduced in Chapter 5.

The order of importance for these workloads is:

major production CICS
support TSO
heavy engineering batch
pilot production CICS
application development TSO
normal production batch
default (open shop) batch

There is not enough expanded storage to accommodate the entire workload. Heavy engineering batch is to be swapped to auxiliary storage. Other paging categories will compete for expanded storage in priority order.

These parameter sets are fairly complete, but the ICS in particular does not have the genuine complexity of an ICS to be found in a real system. Most ICSs are built up over time with numerous layers of exceptions to general workload management rules.

B.6.1. Sample OPT

The management of system resources is defined in the OPT. Workloads are defined by performance groups, and related as appropriate to OPT settings by domains, in the IPS. External work unit names are bound to the IPS by the ICS. We'll examine each of those parameter sets in turn. First, we'll complete the OPT:

```
/*******************      BEGINNING OF OPT        ******************/
/*                          GLOBAL PARAMETERS                       */
MCCMAXSW=576                          /* FIT SWAP PACKET SIZE TO AUX
CONFIG */
SWAPRSF=20                           /* MILDLY RETARD EXCHANGE SWAP*/
MCCAECTH=(150,300)                       /* E.S. STEAL DEFAULTS OK*/
MCCAFCTH=(600,900)                             /*BIG SWAPS LIKELY*/

/ ********************************************************************/
/*                      LOGICAL SWAP CONTROLS                       */
LSCTMTE=(1,5)                        /* DON'T TIE UP CENTRAL FOR LONG
THINKERS
LSCTUCT=(20,30)                       /* DEFAULT UIC CONTROL OR THINK
TIME

/ ********************************************************************/
/*                     MPL ADJUSTMENT CONTROLS                      */
RCCPTRT=(164,205)                    /*STARTING POINT-SEE DISCUSSION*/
RCCCPUT=(106,110)                     /*STARTING POINT FOR FOUR-WAY*/
RCCUICT=(2,4)                            /*START AT SP 4.2 DEFAULT*/

/ ********************************************************************/
/*                     MISCELLANEOUS CONTROLS                       */
CCCSIGUR=45                               /*INITIAL MTTW STEP SIZE*/
CNTCLIST=NO                          /*DEFAULT UNTIL REASON TO CHANGE*/
CPENABLE=(20,40)                      /*TRADE QUEUING FOR LESS DISRUP-
TION*/
DVIO=NO                                 /*DIRECTED VIO IRRELEVANT WITH
ALL-9345s*/
ERV=500                              /*DEFAULT UNTIL PROVEN OTHERWISE*/

/ ********************************************************************/
/*                CRITERIA AGES FOR DEFAULT BATCH                   */
ESCTSWWS(1)=32766                    /* SWAPS TO EXPANDED STORAGE LAST
*/
ESCTSWTC(1)=32766                    /* TRIMS TO EXPANDED STORAGE LAST
*/
ESCTSTC(1)=1000
ESCTPOC(1)=5000
ESCTVIO(1)=5000
ESCTBDS(1)=2000

/ ********************************************************************/
/*           CRITERIA AGES FOR NORMAL PRODUCTION BATCH              */
ESCTSWWS(5)=2000
ESCTSWTC(5)=2000
ESCTSTC(5)=1000
ESCTPOC(5)=2000
ESCTVIO(5)=1500
ESCTBDS(5)=1500                       /* FACILITATE SOME HIPERBATCH IF
SAFE */

/ ********************************************************************/
```

```
/*            CRITERIA AGES FOR HEAVY ENGINEERING BATCH             */
ESCTSWWS(10)=32767                                  /*NO SWAP TO E.S.*/
ESCTSWTC(10)=32767                                  /*NO TRIM TO E.S.*/
ESCTSTC(10)=900                        /*STEALS TO E.S. IF LITTLE MIGRA-
TION*/
ESCTPOC(10)=900                        /*PRIVATE STORAGE MGT A POSSIBIL-
ITY*/
ESCTVIO(10)=32766                                 /*VIO NOT EXPECTED*/
ESCTBDS(10)=32766                                /*HSPACE NOT EXPECTED*/

/ ********************************************************************/
/*                  CRITERIA AGES FOR MAJOR CICS                    */
ESCTSWWS(15)=32766                                    /*NONSWAPPABLE*/
ESCTSWTC(15)=32766                                    /*NONSWAPPABLE*/
ESCTSTC(15)=20                                /*MOST PREFERRED WORKLOAD*/
ESCTPOC(15)=50
ESCTVIO(15)=2000                                  /*VIO NOT EXPECTED*/
ESCTBDS(15)=100                                 /*ALLOW HIPERSPACE LSR*/

/ ********************************************************************/
/*                  CRITERIA AGES FOR PILOT CICS                    */
ESCTSWWS(20)=32767                  /                  *NONSWAPPABLE*/
ESCTSWTC(20)=32767                                    /*NONSWAPPABLE*/
ESCTSTC(20)=150
ESCTPOC(20)=250
ESCTVIO(20)=3000
ESCTBDS(20)=1500                            /*ALLOW SOME HIPERSPACE LSR*/

/ ********************************************************************/
/*                  CRITERIA AGES FOR SUPPORT TSO                   */
ESCTSWWS(25)=100
ESCTSWTC(25)=100
ESCTSTC(25)=30
ESCTPOC(25)=100
ESCTVIO(25)=700
ESCTBDS(25)=1000

/ ********************************************************************/
/*          CRITERIA AGES FOR APPLICATION DEVELOPMENT TSO           */
ESCTSWWS(30)=200
ESCTSWTC(30)=200
ESCTSTC(30)=150
ESCTPOC(30)=200
ESCTVIO(30)=1500

/ ********************************************************************/
/*            CRITERIA AGES FOR MOST-UNLOVED WORKLOADS              */
ESCTSWWS(86)=32766
ESCTSWTC(86)=32766
ESCTSTC(86)=32766
ESCTPOC(86)=32766
ESCTVIO(86)=32766
```

```
/ ********************************************************************/
/*            CRITERIA AGES FOR EMERGENCY REPAIR JOBS                */
ESCTSWWS(99)=5
ESCTSWTC(99)=5
ESCTSTC(99)=5
ESCTPOC(99)=5
ESCTVIO(99)=5
/***********************       END OF OPT       ***********************/
```

B.6.2. Sample IPS

Our sample IPS is built on the "well-ordered CPU" model we introduced in Chapter 5, with the further definition of the workload initially discussed above. Note that dispatching priorities are spread over the entire usable range, with plenty of gaps to allow for future insertions with minimal disruption. If the priorities seem low for some workloads, remember that dispatching priorities are all relative—the numbers don't matter.

```
/************************ BEGINNING OF IPS ***************************/
/*                  SERVICE DEFINITION COEFFICIENTS                  */
CPU=10.0,SRB=10.0,IOC=5.0,MSO=0.0                 /* STAMP OUT MSO! */
/*                         GLOBAL PARAMETERS                         */
IOQ=PRTY                              /* ENABLE PRIORITY I/O QUEUING */
TSPTRN=(1,2,1,*)                      /* SIMPLE PATTERN IF T-SLICE USED
*/
/*              IN THIS IPS, JES2 IS AT HIGH DP 50%,                 */
/*             DFHSM IS PREFERRED 25%, AND NO TSDP IS                */
/*                     ACTIVE 25% OF THE TIME                        */
PVLDP=F2                              /* JUST ABOVE SIGNIFICANT TSO 2ND
PERIOD */
IOSRVC=COUNT                                  /* PAY FOR INEFFICIENCY */
/************************       DOMAINS        ************************/
/*        DOMAIN 1 IS NEEDED FOR OTHERWISE UNCONTROLLED BATCH        */
DMN=1,CNSTR=(0,50),DSRV=(5000,10000)                  /* DFLT BATCH */
/*             DEFAULT TSO HAS 150 LOGONS AT PEAK TIMES              */
DMN=2,CNSTR=(30,50),DSRV=(100000,100000)          /*DEFAULT TSO PER 1*/
DMN=3,CNSTR=(5,20),ASRV=(2000,5000)               /*DEFAULT TSO PER 2*/
DMN=4,CNSTR=(0,20),DSRV=(5000,20000)              /*DEFAULT TSO PER 3*/
/*              SUPPORT TSO HAS 15 LOGONS AT PEAK TIMES              */
DMN=5,CNSTR=(3,15),DSRV=(500000,500000),ESCRTABX=25      /* PERIOD 1 */
DMN=6,CNSTR=(1,15),ASRV=(5000,10000),ESCRTABX=25         /* PERIOD 2 */
DMN=7,CNSTR=(1,15),DSRV=(10000,50000),ESCRTABX=25        /* PERIOD 3 */
/*              APP DEV TSO HAS 50 LOGONS AT PEAK TIMES              */
DMN=8,CNSTR=(10,20),DSRV=(300000,300000),ESCRTABX=30     /* PERIOD 1 */
DMN=9,CNSTR=(2,10),ASRV=(4000,8000),ESCRTABX=30          /* PERIOD 2 */
DMN=10,CNSTR=(0,10),DSRV=(7500,30000)        /*PERIOD 3, DEFAULT ESCT*/
/*             SINGLE PERIOD FOR HEAVY ENGINEERING BATCH             */
DMN=11,CNSTR=(1,5),DSRV=(10000,300000),ESCRTABX=10
/*                    PRODUCTION BATCH, 2 PERIODS                    */
DMN=12,CNSTR=(1,30),DSRV=(3000,30000),ESCRTABX=5
```

```
DMN=14,CNSTR=(0,30),DSRV=(10000,300000),ESCRTABX=5
/*       SWAP-OUT DOMAIN (TRADITIONAL NUMBER BREAKS SEQUENCE)       */
DMN=13,CNSTR=(0,0),ESCRTABX=86
/*                        DEFAULT STARTED TASKS                      */
DMN=15
/*                           PREFERRED CICS                          */
DMN=16,ESCRTABX=15                                        /* TOR */
DMN=17,ESCRTABX=15                                     /* AOR/ROR */
/*                            ORDINARY CICS                          */
DMN=18,ESCRTABX=20                                        /* TOR */
DMN=19,ESCRTABX=20                                     /* AOR/ROR */
/*                     "SPONGE" FOR FOUR PROCESSORS                  */
DMN=20,CNSTR=(4,4),ESCRTABX=86
/*                    EMERGENCY RED-HOT REPAIR JOBS                  */
DMN=99,CNSTR=(2,2),ESCRTABX=99
/******************        PERFORMANCE GROUPS       ******************/
/******************          BASIC DEFAULTS         ******************/
PGN=1,(DMN=1,DP=M1)                    /* DEFAULT BATCH AS REQUIRED */
PGN=2,                                  /* DEFAULT TSO AND APPC/MVS */
 (DMN=2,DP=F33,DUR=300)                 /* DUR PROVIDES 85% IN PER 1*/
 (DMN=3,DP=M2,DUR=800)                           /* 12% IN PERIOD 2 */
 (DMN=4,DP=M1)                                     /* ALL THE REST */
/******************         SIGNIFICANT TSO         ******************/
PGN=3,                                             /* SUPPORT TSO */
 (DMN=5,DP=F54,DUR=300)
 (DMN=6,DP=F34,DUR=800)
 (DMN=7,DP=M3)
PGN=4,                          /* APPLICATION DEVELOPMENT TSO */
 (DMN=8,DP=F52,DUR=900)
 (DMN=9,DP=M2,DUR=3000)
 (DMN=10,DP=M1)
/******************       DISTINGUISHED BATCH       ******************/
PGN=5,(DMN=11,DP=M2)                            /* HEAVY ENGRG BATCH */
PGN=6,                                /* PRODUCTION BATCH, 2 PERFORMANCE
PERIODS */
 (DMN=12,DP=M2,DUR=20000)                      /* CLASS Q (1 CPU SEC) */
 (DMN=14,DP=M1)
/*********************       APPC/MVS TP'S       ***********************/
PGN=7,(DMN=8,DP=F6)                             /* APPC/MVS HOT TXNS */
PGN=8,(DMN=8,DP=F51)                              /* APPC/MVS E-MAIL */
/******************         STARTED TASKS          ******************/
PGN=10,(DMN=15,DP=M1)                         /* DEFAULT NONSWAPPABLES */
PGN=11,(DMN=16,DP=F70)                              /* MAJOR CICS TOR */
PGN=12,(DMN=17,DP=F43)                           /* MAJOR CICS AORS/RORS */
PGN=13,(DMN=18,DP=F24)                              /* MINOR CICS TOR */
PGN=14,(DMN=19,DP=F22)                           /* MINOR CICS AORS/RORS */
/****************       SYSTEM ADDRESS SPACES      ******************/
PGN=20,(DMN=15,DP=F92)                    /*DIRECT-CONNECT REAL-TIME MONI-
TOR*/
PGN=21,(DMN=15,DP=F84)                            /* HISTORICAL MONITOR */
PGN=22,(DMN=15,DP=F82)                                    /* VTAM */
PGN=23,(DMN=15,DP=F80)                               /* VTAM MONITOR */
```

```
PGN=24,(DMN=15,DP=M8)                              /* HIGH-PRIORITY STCS */
/*              PGN 24 INCLUDES SMS, APPC, AND ASCH                      */
PGN=25,(DMN=15,DP=F74)                           /* DB2 MONITOR */
PGN=26,(DMN=15,DP=F72)                              /* DB2 IRLM */
PGN=27,(DMN=15,DP=F71)                          /* CICS MONITOR */
PGN=28,(DMN=15,DP=F64)                              /* DB2 MSTR */
PGN=29,(DMN=15,DP=F62)                              /* DB2 DBM1 */
PGN=30,(DMN=15,DP=F60)                           /* LLA AND VLF */
PGN=31,(DMN=15,DP=F50)                              /* DB2 DIST */
PGN=32,(DMN=15,TSDP=F44,DP=F30,TSGRP=1)                 /* JES2 */
PGN=33,(DMN=15,TSDP=F42,DP=F02,TSGRP=2)                /* DFHSM */
PGN=34,(DMN=15,DP=F40)                    /* RMF WRITER ADDRESS SPACE */
PGN=35,(DMN=15,DP=F04)                  /* CONFIGURATION/CHANGE MONITOR*/
/***************       SPECIAL PERFORMANCE GROUPS        **************/
PGN=86,(DMN=13)                            /* UNCONDITIONAL SWAP-OUT */
PGN=87,(DMN=1,DP=F0)                                /* BOTTOM FEEDER */
PGN=88,(DMN=13,DUR=900,UNT=R)             /* 15-MINUTE SWAP-OUT,THEN */
 (DMN=1,DP=M1)                                      /* SAME AS PGN 1 */
PGN=89,(DMN=20,DP=M0)                    /* SPONGE FOR EXCESS CYCLES */
PGN=99,(DMN=99,DP=F94)                 /* SUPER-EMERGENCY WORK ONLY!!!!*/
/**********************       END OF IPS       ***********************/
```

B.6.3. Sample ICS

Now that we have a moderately well-commented IPS, building the ICS is simply a matter of translating the identifying comments into the syntax of the ICS. By doing so, we "translate" the names by which work units are known in the outside world to the simple system of performance group numbers. Neither report performance groups nor optional performance groups are defined in this ICS. RPGNs are not essential to the purpose of the example, and the conditions for self-policing use of OPGNs do not exist in the accompanying IPS.

```
/*************************  BEGINNING OF ICS  ***********************/
MASK=*
SUBSYS=STC,PGN=10
 TRXNAME=ASCH,PGN=24
 TRXNAME=APPC,PGN=24
 TRXNAME=SMS,PGN=24
 TRXNAME=VLF,PGN=30
 TRXNAME=LLA,PGN=30
 TRXNAME=NET,PGN=22
 TRXNAME=NETMON,PGN=23
 TRXNAME=REALMON,PGN=20
 TRXNAME=HISTMON,PGN=21
 TRXNAME=DB2MON,PGN=25
 TRXNAME=DB2IRLM,PGN=26
 TRXNAME=CICSMON,PGN=27
 TRXNAME=DB2MSTR,PGN=28
```

```
  TRXNAME=DB2DBM1,PGN=29
  TRXNAME=DB2DIST,PGN=31
  TRXNAME=RMF,PGN=34
  TRXNAME=CFIGMON,PGN=35
  TRXNAME=CICSPT(1),PGN=11
  TRXNAME=CICSPR(1),PGN=12
  TRXNAME=CICSPA(1),PGN=12
  TRXNAME=CICSXT(1),PGN=13
  TRXNAME=CICSXR(1),PGN=14
  TRXNAME=CICSXA(1),PGN=14
SUBSYS=JES2,PGN=1
  TRXCLASS=E,PGN=5
  TRXCLASS=M,PGN=6
  TRXCLASS=N,PGN=6
  TRXCLASS=O,PGN=6
  TRXCLASS=P,PGN=6
  TRXCLASS=Q,PGN=6
  TRXCLASS=T,PGN=6
SUBSYS=TSO,PGN=2
  USERID=ADEV*,PGN=4
  USERID=TSUP(1),PGN=3
SUBSYS=ASCH,PGN=10
  TRXCLASS=HOT,PGN=7
  TRXNAME=E_MAIL,PGN=8
/**********************        END OF ICS        **********************/
```

B.7. Summary

The set of SRM parameters in the OPT and IPS members of SYS1.PARMLIB, and the workload assignment information in the ICS member, are the means of controlling how MVS in compatibility mode deals with its resources and workloads. The history of MVS, embedded in the defaults of OPT and IPS, had in the past caused inconsistent and often counterproductive system operation. Although the defaults have improved in recent MVS/ESA releases, better workload-oriented controls demand evaluation and customization. An essential step in performance management is to take control of the system away from the software vendor and place it firmly with the installation that paid for it.

Information provided by IBM about these parameter sets and their individual elements is factual and nonjudgmental. In this and the preceding appendix, we have developed a more evaluative view of how the SRM can and should be controlled. Where advice is offered, justification and explanation accompanies it. Readers may draw their own conclusions about the

worth of the advice, but they are urged to try it before rejecting it.

B.8. Chapter Questions

1. Review your dispatching priority layout. Determine if the more important workloads have consistent treatment in your IPS in all key resource areas—are they favored to the same extent in dispatching priority, MPL, and expanded storage eligibility? Correct the IPS as required.
2. Check the dispatching priority relationships in your IPS. Are servers above the address spaces they serve? For example, is JES2 low enough in priority to avoid interference with first period TSO, but high enough to avoid having devices or lines time out?
3. Examine the use of performance periods in your IPS. Are they structured according to the considerations discussed above? As a transaction moves to the next period, is it likely to be swapped out only because the new domain's constraints are too limited? Why? If it's broke, fix it!
4. If you have no ICS, create one. If you have one, check it for completeness (all work captured) and excess complexity, such as multiple OPGNs or OPGNs without self-policing tradeoffs.
5. Adapt the sample parameter sets on the preceding pages to your system. How do they compare with what you've been using to date? Does an alternate viewpoint help you to get a better understanding of your system?

Glossary

Author's note: The definitions in this glossary are those assumed in the text and are my own; they are reasonably consistent with standard definitions. Terms in this glossary are in general those not contextually defined in the text or those for which an extended discussion seemed necessary. For a term not found here, the first reference found in the index should supply a contextual definition. Another book in this series, *MVS: Concepts and Facilities*, by Robert H. Johnson, also has an extensive glossary with useful alternative points of view for some terms. Cross-references within the glossary are shown in **bold** type.

The ultimate reference for IBM's view of SRM-related terms and functions is the current edition of *Initialization and Tuning* for the system under consideration. For Workload Manager goal mode, it is the corresponding edition of *Planning:Workload Management*.

A few definitions were adapted from those in *Computer Desktop Encyclopedia*, reprinted with permission. ©1997 The Computer Language Co. Inc.

[Special Characters]

* (asterisk) used in storage isolation specifications to indicate maximum value; used in objectives to signify linear interpolation; the latter use is obsolete in MVS/ESA SP 4.2 and later releases.

A

ABEND ABnormal END, the condition that occurs when a program terminates by issuing the ABEND **SUPERVISOR CALL** (**SVC** 13), or is terminated because of an unexpected program check or other condition that MVS cannot handle. A common consequence of an ABEND is a dump.

above the line addresses above 16 megabytes. In MVS/370 the term meant real storage in the extended addressing range; in MVS/XA and MVS/ESA it refers to the extended virtual areas. The significance of "above the line" in MVS/370 was the need to move I/O buffers "below the line" before START I/O [FAST] could be issued. In XA and ESA, virtual storage below the line is the area threatened by virtual storage constraint; moving code and data above the line by making program

changes is the way to relieve such constraint. In XA and ESA, central storage having absolute addresses below 16 MB is still needed to support I/O using Format-0 CCWs.

absolute address the lowest level of central storage addressing in a processor complex in basic mode, or within a single logical partition. An absolute address is not adjusted by prefixing or translated by dynamic address translation, and is the object of a CCW address. There is an unnamed lower level of addressing in logically partitioned mode; only one partition can have "logical" absolute addresses corresponding to "physical" absolute addresses.

absolute storage central storage accessed through absolute addressing.

access control system system designed to define, control, and report on [attempted] access of users (jobs, TSO sessions, individual users of transaction processing systems) to resources, most often data sets, transactions, terminals, or other entities defined to the access control system. Examples are IBM's RACF and Computer Associates' ACF2 and Top Secret.

access method part of an operating system providing the interface and device-driving code between high-level I/O requests (e.g., GET, PUT, READ, WRITE) and I/O devices as seen by the I/O supervisor (IOS). Access methods translate logical record requests to the physical data locations, create channel programs and convey them to IOS for execution, manage retries of unsuccessful I/O, and manage queued requests and buffers.

access register in ESA, a register activated in access register (AR) mode to supply (indirectly) the segment table origin for data operands paired with the same-numbered general register used as a base register. The general register supplies the base address within the data space or address space designated by the segment table found by means of the access register.

account number a character string assigned to an address space by the positional "account" field in a JOB statement, in the TSO UADS, or in an equivalent source maintained by an access control system. In MVS/XA and MVS/ESA systems, the account number or a substring of it may be used as a basis for performance group assignment in the ICS. In Workload Manager goal mode, the accounting data may be used for classification of work units to service classes.

accounting data a more precise designation for **account number**.

active I/O an execution state in which a workload is waiting for completion of an input or output operation while not not having another

dispatchable task. Active I/O is usually considered to be a productive state, but it also represents a tuning opportunity, since most sequential I/O and much random I/O can and should be overlapped with instruction execution.

activity rate a measure of throughput in work units per second. Tracking throughput is one of the essential functions of performance management.

actuator mechanism of a direct access storage device (DASD) that moves the arm, which in turn carries the read and write recording heads and associated electronics; by extension, the arm itself and the assembly of arm[s] and disk[s] that corresponds to a single device address or device number in a DASD subsystem.

Actuator-Level Buffering use of a small cache or buffer holding a track's worth of data to eliminate variable rotational delay (RPS miss) in DASD. The term is a trademark of Storage Technology Corporation. The technique is an alternative to buffering in the storage director or control unit, as in the IBM 3990 and 9343 control units.

adaptive resource management management of resources in a computer system based on the behavior of the work that runs in the system and goals for the performance of that work, as opposed to management based on parameters that specify how the resources are to be apportioned.

address space a linear range of virtual storage addresses from 0 through the architecturally defined maximum for a given system. Each work unit (job, started task, TSO session, subsystem) is synonymous with at least one address space. An address space in MVS is defined by a segment table, its associated page tables, and several control blocks, notably the ASCB. It contains a common area and a private area. Address space layouts may be found in Chapter 3.

addressing mode the means by which the address parts of an instruction (base, displacement, optional index) are combined with other information to yield an absolute address. Absolute addresses are direct references to absolute storage. Real addresses may be altered by prefixing before being treated as absolute. Virtual addresses are translated to real addresses by dynamic address translation. Cross-memory mode uses a secondary segment table origin to initiate dynamic address translation for operands of certain instructions. Access register mode uses access registers to dynamically select segment table origins for each operand reference accessed through a base register. The final choice, independent of the others, is between 24-bit and 31-bit address generation.

ADSL Asymmetric Digital Subscriber Line (ADSL), a new modem technology that converts existing twisted-pair telephone lines into access paths for multimedia and high speed data communications. ADSL transmits more than 6 Mbps to a subscriber, and as much as 640 kbps more in both directions. Such rates expand existing access capacity by a factor of 50 or more without new cabling. ADSL can transform the existing public information network from one limited to voice, text and low resolution graphics to a system capable of bringing multimedia, including full motion video, to subscribers' homes.

AFC see **AFQ**.

AFQ available frame queue; sometimes AFC for available frame count. The queue is the ordered set of frames available for assignment by the Real Storage Manager; the count is the number of such frames in the queue.

AIX/6000 AIX/6000 is the IBM variant of UNIX that runs on the RS/6000 platform.

ALB see **Actuator-Level Buffering**.

alert an exception message, particularly one that is displayed on an operator's console.

allegiance set of electronic states denoting the association of channel, control unit, [device controller,] and device for the duration of an I/O operation. Allegiance is established during unoverlapped protocol time and disbanded during overlapped protocol time.

ALLOCATE command TSO equivalent of the JCL DD statement; completes the late binding of DD names in program references to physical I/O resources, usually the TSO terminal or data sets on DASD.

allocation set of operating system services to make associations between programs and I/O resources. Allocation establishes the required connections when specific resources (such as catalogued data sets) are needed, and additionally selects the resources to be assigned when the request is nonspecific, as for a temporary data set or a tape drive.

allocation recovery series of console messages and responses when the I/O resources called for by a program (usually a batch job step) are not all available. A series of device numbers is presented to the operator, who is expected to make a selection from among them. When allocation recovery is invoked for other than tape units being switched between systems, a missing data set is often the cause and the operator dialogue is unlikely to succeed.

alternate path a redundant I/O connection from device through [device controller,] control unit, and channel to a CPU. In MVS/370, alternate paths provided enhanced availability at the cost of performance deg-

radation. In XA and ESA, DASD supporting dynamic pathing provides benefits in both availability and performance from dynamic pathing. Up to four active paths are supported in current device subsystems.

analytic model a mathematical representation of a physical system (in this case a computer under MVS) that takes account of processing times, delays, and queues in accounting for or predicting the performance of the physical system. Contrast with **simulation model**.

AOBJ (obsolete as of MVS/ESA SP 4.2) the use of an objective to control the MPL of a domain, in which the average service rate received by each address space in the domain is the independent variable of the objective.

APF see **authorized program facility**.

APG automatic priority group, an obsolete IPS parameter to set the dispatching priority within the original restricted MVS APG. An APG specification in IEASYSxx sets the high-order hexadecimal digit of the priority; the APG specification in the performance period in the IPS sets the low-order digit. The use of the APG parameter is incompatible with current SRM facilities for controlling dispatching priority.

APG range range of high-order hexadecimal digits of true dispatching priority subject to assignment by the DP, IOP, PVLDP, and TSDP IPS keywords. The range and its implicit mapping are set by the APGRNG keyword.

APGRNG IPS keyword to set the **APG range** controlled by the IPS. This parameter was dropped as of MVS/ESA Version 5 compatibility mode.

API application programming interface, a set of rules and conventions, possibly including service routines and macro definitions, that allow independent programs to communicate with the program "owning" the API. An access method is an example of an API.

APPC/MVS set of services for peer-to-peer or client-server communications between transactions running under control of the APPC subsystem in MVS and transactions on other platforms or server address spaces on MVS, or any combination of such elements. APPC is one of the principal workload subsystems in the ICS of compatibility mode or in goal mode classifications.

applet in Microsoft Windows, a small application subordinate to a major function such as the set of Control Panels; in Sun Microsystems' JAVA environment, a JAVA application downloaded from a server and executed on the client's JAVA Virtual Machine under control of the JAVA interpreter or compiler. Applets in JAVA are not limited in size and may be substantial applications.

application programming interface see **API**.

AR access register; also the **addressing mode** that makes use of access registers.

architecture the highest level of logical design for a computing system or family of such systems, defining such elements as the instruction set, the objects operated on by the instructions, exceptions and their manifestation as interrupts, and reserved facilities. An architecture usually spans more than a single product family and may include features or facilities to be released at a future time. Current engineering designs may thus contemplate extendibility to a fuller implementation in the future.

archive to copy data for safekeeping and possible future use for recovery to another physical embodiment, usually from DASD to tape.

array data aggregate in which each element is of the same type and size.

array storage layout representation in storage of an array, the essential matter of concern being the order in which elements are stored. If the order of use or reference approximates the order of storage, the working set of the program making use of the array is minimized.

ASCB address space control block, the principal MVS data area defining the attributes of an address space.

ASCII American Standard Code for Information Interchange—a 7- or 8-bit set of character encodings defining the information content of text data bytes. ASCII is the standard for alphabetic languages except in IBM systems derived from System/360, which use **EBCDIC.**

ASID address space identifier, a number denoting the position of a pointer to an address space's ASCB in an array called the ASVT (address space vector table). The ASVT and ASID are architecturally defined in the various levels of System/370, XA, and ESA so that microcode-assisted routines may aid in speeding up key MVS functions that manipulate ASCBs.

ASM Auxiliary Storage Manager, the MVS component responsible for controlling and initiating I/O to and from page and swap data sets.

ASM queue queue of pending I/O requests managed by the ASM.

ASN address space number, an engineer's or architect's synonym for **ASID**.

ASRV IPS parameter specifying the minimum and maximum service rates corresponding to contention index values of 100 and 1, respectively, for a domain. The contention index is determined based on the average service rate of all address spaces in the domain.

ATTACH the MVS service, or its invoking macro-instruction or SVC, that creates new tasks.

authorized program facility (APF) the means for designating libraries as eligible to contain authorized programs which are granted the authorized status during execution. A program is designated as authorized by setting the AC bit in its directory entry to 1, usually by action of the Linkage Editor. An authorized program fetched from an authorized library is permitted to issue otherwise restricted SVCs, in turn giving it the ability to gain privileged states and storage-protect keys, giving it access to any part of the system.

automated operations replacement of human operator activities with programmed operations, with the purpose of eliminating repetitive tasks, performing selected activities without human intervention, simplifying complex tasks, and interpreting the message stream and reducing its volume, thereby enabling the operator to make constrained decisions and reducing the likelihood of operator error.

automatic priority group see **APG**.

auxiliary storage the set of page and swap data sets.

Auxiliary Storage Manager see **ASM**.

auxiliary storage shortage condition detected by the SRM in which the supply of unused auxiliary storage slots falls below a predetermined threshold.

auxiliary storage shortage swap swap-out initiated by the SRM in response to the free slot count being lower than a threshold value.

auxiliary storage slot unit of measure of auxiliary storage; a slot is the same size as a frame or a page, 4096 bytes.

availability portion (usually expressed as a percentage) of scheduled service time during which a computing system or service is fully usable by its customers. Tracking availability is one of the essential functions of performance management.

available frame count see **AFQ**.

available frame queue see **AFQ**.

B

b symbol or abbreviation for bit(s)

B symbol of abbreviation for byte(s)

back-end completion stage of a process that is partially asynchronous. Back-end I/O processing, for example, occurs in the I/O sec-

ond-level interrupt handler after the channel-end and device-end interrupts are received by the CPU.

backing real resources used to materialize virtual resources; pages are backed by frames of real storage. The term has a somewhat different meaning in real storage hierarchies—the higher-level, smaller, faster, more expensive storage element is fully backed by the slower, cheaper, and more abundant lower level. In some cache I/O subsystems, such as the IBM 3990 Model 3 with the Fast Write feature, nonvolatile storage at the higher level is used to defer backing.

backup jobs batch jobs used to preserve the content of essential data resources in archives.

balanced methodology performance management approach making optimum use of service data, resource data, bottleneck data, and contention data.

balanced system system configuration with adequate but not excessive resource in all key hardware areas: CPU, central storage, expanded storage, channels, control units, and devices. A balanced system when loaded to capacity will show a similar degree of constraint in all resource areas. The term (but not the definition) was originated and popularized by Ray Wicks of the IBM Washington Systems Center.

Base Control Program that part of an MVS system other than the JES and other independently controlled subsystems.

batch workload consisting of discrete *jobs*, managed up to initiation and after termination by the JES.

Batch Message Processor batch job making use of IMS data base facilities, often used for report creation, mass inquiries, or bulk updates to data bases.

batch window time period during which essential batch jobs must be completed. Such jobs are often not compatible with online service, either by demanding exclusive access to critical resources or by causing such high activity rates that online service would be unacceptably slow.

BatchPipes optional (priced) facility for MVS systems starting with SP4.3 that enables sequential data to be transmitted from one job to a subsequent job using a FIFO queue in a data space rather than a data set that will be deleted after it is read. Such a data set is often called a transient data set. BatchPipes alone is now obsolete, having been superseded by **SmartBatch**, a more comprehensive set of service.

below the line see **above the line**

BCP see **Base Control Program**.

binary search search technique in which the ordered list denoting the items to be searched is successively halved, discarding the half not encompassing the item's identifier. The technique is efficient if the list contains only identifiers; if the items are large and constitute the list that is searched, the principle of validity of reference is violated, and working-set size (for internal searches) or I/O time (for external searches) may be excessive.

binding process of resolving symbolic references to data by associating physical resources with the references. *Early* binding is efficient but limits flexibility; *late* binding requires more effort and more formally defined interfaces but preserves maximum flexibility.

bipolar semiconductor technology that requires continuous power to maintain the binary state of each switching element. The power demand makes this technology run hot, requiring high-volume forced-air or circulating water for cooling. Bipolar has until recently been capable of faster switching speeds than **CMOS** technology; its need for specialized cooling and lower circuit density make it much more expensive than CMOS and thus likely to become obsolete as soon as CMOS attains an undisputed lead in circuit speed.

block multiplexing I/O technique in which a portion of a path (notably the channel) may be processing more than one request at a time. Only one request at a time may be connected through the entire path; other requests are disconnected. In block multiplexing, data transfer proceeds without interruption. Byte multiplexing is a lower-speed technique in which the data stream is shared among active low-speed devices.

block multiplexor channel channel capable of block multiplexing.

block paging alternate name for paging using the contiguous-slot allocation technique, extended in SP 4.2 to include blocked page stealing, faulting, and incremental swapping.

block size size in bytes of the unit of physical data transfer in an I/O operation.

BMP see **Batch Message Processor**.

bottleneck analysis performance monitoring technique that identifies execution states of a workload, leading to a table or histogram of frequency of each state. A bottleneck is a state that prevents the workload from achieving its service goal; identification and easing of bottlenecks is a key part of performance management. Use of this technique is an essential part of Workload Manager goal mode.

buffer area of storage that contains a physical record or block. Optimum use of buffers is a trade-off between I/O avoidance and excessive

use of virtual storage. Buffers use fixed real storage during I/O operations.

bus-and-tag channels another way of describing parallel channels, focusing on the large and heavy cables used to attach devices and control units to the channels. One cable set (bus) carries the data; the other (tag) carries addressing and control information.

byte smallest unit of addressable data in central storage. In current IBM architecture, a byte is 8 bits.

C

C (programming language) Low-level, block-structured programing language originally developed as the implementation language for UNIX, now available on most platforms. An object-oriented extension to the language, C++, is eclipsing the original C language in popularity.

cache in a storage hierarchy, a small unit of fast storage that contains the active data elements otherwise found in a larger unit of slower (less expensive) storage. Examples of caches include the high-speed buffer associated with a CPU and the storage in a cached DASD control unit.

cache control unit an I/O control unit with a cache. Tape and DASD control units are most often augmented with caches.

cache controller see **cache control unit**.

cache device strictly, an I/O storage device with a built-in cache; in common usage, a device connected to a channel through a cache control unit.

CADAM Computer-aided design and manufacturing, a subsystem for very precise engineering and manufacturing graphics, originally developed by Lockheed but now marketed by IBM as one of a number of such high-function graphics applications. CADAM was originally implemented on OS/360 and has been slowly updated to current MVS levels and to run on other platforms.

capacity management art of planning and implementing sufficient (but not excessive) computing capacity to enable an establishment's data processing needs to be met. See also **balanced system**.

capture ratio ratio of CPU use which can be accounted for in CPU (TCB and SRB) service accumulation, to absolute CPU use in the same time interval. Uncaptured time is often accounted for by multiplying captured service by the reciprocal of the capture ratio. Capture ratios differ by types of workload, so such a correction may not be equitable.

CAT see **Criteria Age Table**.

catalog data set containing at minimum the locations (volume IDs) of other data sets. For VSAM data sets, the catalog also contains comprehensive information about the data set. Catalogs also contain alias pointers to other catalogs, which in turn contain entries for data sets whose names begin with the string that forms an alias pointer. Active catalogs form a hierarchy topped by one master catalog.

catalogued data set that may be found through a search of catalogs beginning with the master catalog.

catalogued procedure misnomer for a set of predefined JCL statements contained in a library (partitioned data set) known to the JES. The library need not be a catalogued data set since it is allocated to JES at startup time. A catalogued procedure is invoked by simply specifying its name in a JCL EXEC statement. If the procedure contains symbolic parameters, the actual parameters to be substituted are supplied in the invoking JCL.

CAW Channel Address Word.

CCW see **channel command word**.

CEC **central electronic complex**, now usually known as a processor complex. "CEC" (pronounced "kek") is the preferred designation for the central set of elements in a CMOS system such as the IBM 9672.

central electronic complex the set of CPUs, channels, and auxiliary elements that constitute the resources of a computing system other than channel-attached I/O devices.

central storage real storage in which instructions and data reside. The term was introduced when expanded storage was introduced, to denote the subset of real storage that is not expanded storage.

chained searching search technique in which data elements are unordered but have identifiers that are in a sequence linked by pointers to the next element and optionally to the preceding element. Chained structures are easy to build with minimum data rearrangement but often expensive to search. For large chained structures, a preliminary search through a "thumb index" can restrict the search to only a portion of the structure. Sequential searching through a large chained structure violates the principles of validity of reference and minimum working set size.

channel part of a computer system that performs input/output operations, linking central storage with I/O devices, as specified by channel command words. An I/O operation is initiated by a CPU instruction, which designates the starting point for the channel operation to commence. The channel then proceeds independently, asynchronously with the CPU, until its sequence of command words ends. When the

last I/O operation of the sequence ends, the channel causes an interrupt in the CPU. Even though a channel may not be fully independent of the CPU (as in the cycle-stealing channels of the IBM 4381), the series of operations is conceptually followed.

channel command word (CCW) control word similar to an instruction, designating an I/O operation to be performed, the data address in absolute storage that is to be used, the number of bytes to be transferred, and bits designating options to be invoked for the operation.

channel measurement block architecturally defined interface to the channel subsystem in XA and ESA to enable reporting programs to extract information on I/O activity.

channel program series of CCWs defining a complete logical sequence of I/O operations and linked together by means of the command-chaining option bit in each CCW but the last.

channel-to-channel adapter (CTC) a pseudo-device that enables two CECs to be connected together using a channel of each complex. Systems connected this way are sometimes known as loosely coupled multiprocessors. CTC connections are fast and efficient, but can cause high channel utilizations, leading to delay of other I/O using that channel.

CI contention index in compatibility mode; also, in VSAM, control interval.

CICS/ESA Current name for IBM's Customer Information Control System, a very popular transaction-processing subsystem.

CIO Chief Information Officer, the executive in charge of all computer-related operations and services in a corporation.

CISC "complex instruction-set computer," a somewhat pejorative back-formation acronym for the type of computer architecture that is not **RISC**.

CKD see **count-key-data**.

client-server loosely, a multi-level computing architecture in which dependent client systems (usually at workstations or personal computers) are dependent on (usually) larger server systems for files, applications, or data. A client-server application is typically more structured, with programming interfaces extending across the connections between client and server.

CMOS complementary metal-oxide semiconductor, a circuit technology in which swtching states are maintained without the continuous application of electrical power. Consequently, CMOS runs cooler than its rival, **bipolar**, enabling high circuit density and relatively low cost.

CMOS has lagged bipolar in speed but appears to be catching up and should soon pass bipolar's speed based on CMOS's higher density and smaller components.

coincident usage an imprecise approach to contention analysis in which a resource in contention is identified, followed by identification of its users in the time span of interest. Within that set of users is the one causing the contention. The main value of this approach is to rule out noncontenders.

common area the area of virtual storage whose segments are in each address space's segment table, and thus addressable from any address space.

common page-in delay delay state of an address space waiting for resolution of a page fault in the common area, from either the PLPA or CSA page data set.

compaction method of reducing storage needed for data based on the content of the data, such as substituting codes and counts for repeated characters or repeated groups of characters.

compatibility mode option as of MVS/ESA SP Version 5 to use the pre-existing SRM controls rather than Workload Manager goal mode. By extension, any MVS system not running in goal mode.

compression method of reducing storage needed for data based on collapsing redundancy in the coded representations of data.

connect time that portion of the time consumed by an I/O operation during which control information and data are transmitted over the path established for the operation. In a DASD I/O this includes a portion of protocol time as well as search and data transfer time.

console automation see **automated operations**.

contention analysis performance analysis technique in which the focus is on a workload of interest, and the data gathered identifies other workloads using resources for which the workload of interest is waiting.

contiguous-slot allocation the use of a local page data set for page-out of a group of related pages (such as a swap-out group) assigned to contiguous slot locations within a single cylinder of the data set. "Contiguous slot," sometimes called block paging, was introduced as a Small Programming Enhancement (SPE) prior to the release of MVS/SP 1.3, then incorporated into that release. Contiguous slot makes the channel programs of local page data sets as efficient as those of swap data sets; for 3380 devices they are more efficient. The combination of contiguous slot and extended swap has all but made swap data sets obsolete.

control register one of a set of special registers included in every IBM system architecture since the introduction of System/370. Control registers contain control bits denoting the modal states of certain features (as a logical extension to the Program Status Word), mask bits selecting which of certain system elements are active in a particular context, ASIDs, and addresses. Control registers are not visible to problem-state programs. Some are loaded explicitly with the LOAD CONTROL instruction and others are loaded implicitly as selected conditions occur during normal program execution. In general, control registers are of no significance to ordinary (problem-state) programs; their content may be inspected using the STORE CONTROL instruction, itself a privileged (supervisor state) instruction.

controlled variable in MVS, a measurable quantity that is controlled by SRM. Such quantities as CPU utilization, page fault rate, available frame count, and unreferenced interval count (UIC) are subject to SRM control. The value of a controlled variable is worthless as an indicator of system activity or capacity when it is within a controlled range. If, for instance, MPL adjustment tends to keep CPU utilization between 95 and 99 percent on average, a utilization of 85 percent indicates a surplus of CPU capacity, but a utilization of 97 percent simply means that MPL adjustment is effective. In Workload Manager goal mode, the control is over the performance of the workloads and utilization values may once again be valuable as indicators of system loading.

count-key-data IBM's basic large-system DASD architecture, in which data tracks have no intrinsic format, but are formatted to reflect actual data stored. Each block or physical record is of variable length and consists of a *count* field identifying the record and specifying its size, an optional *key* field, and a *data* field, with gaps between the fields and following the block. In contrast, DASD in smaller systems uses a newer fixed-block architecture (FBA) in which the device is pre-formatted in fixed-length blocks much as are hard disks in personal computers.

Coupling Facility a processor complex without I/O channels but with high-speed fiber links to the MVS images in a parallel sysplex. The coupling facility, running under control of specialized object-oriented programming is used to facilitate data sharing by holding control information and data buffers in specific structures with the designations of list, lock, and cache. Most MVS subsystems use the coupling facility to the extent that it can contribute to enhanced integrity and performance.

CPU delay state of an address space ready to use a CPU but not sufficiently high on the dispatching queue to receive service in a measurement interval or sample.

CPU overcommitment having more address spaces in the multiprogramming set than the CPU[s] can service. If central storage can accommodate all such address spaces with acceptably low paging delay, the lowest-priority address spaces can receive service when a sufficient number of higher-priority address spaces are even temporarily nondispatchable. If CPU overcommitment can be tolerated, throughput can be maximized. The current fair access dispatcher design can enable more address spaces to receive service than the previous first-ready first-served could do.

CPU queuing CPU delay, usually as described in an analytic model.

CPU, well-ordered a system in which address spaces with high dispatching priority use relatively short bursts of CPU service, with the length of the service burst increasing as priority decreases. This ideal picture is rarely attained and is compromised by the high-priority global SRB activity engendered by I/O completions.

criteria [*sic*] age criterion used to determine eligibility of a class of page for movement to expanded storage. Current migration age is always considered, augmented in certain categories by UIC or think time. The criteria age function is not visible in goal mode.

Criteria Age Table table containing all criteria ages.

cross-memory addressing see **addressing mode**.

cross-memory mode attribute of an address space making use of cross-memory services, or the target of such services, or using secondary addressing mode.

cross-memory services set of MVS services enabling use of the PROGRAM CALL and PROGRAM TRANSFER instructions to allow address spaces to invoke, synchronously, functional code in other address spaces.

CSA common service area or common storage area, the common area.

CSA page data set the second-named page data set in an MVS system. It is used only for page-outs of common-area pages. PLPA pages are not included unless the PLPA page data set is too small to contain all such pages.

CTC see **channel-to-channel adapter**.

CTC ring configuration of CTC connections supporting the GRS function across multiple MVS systems. As of OS/390 Release 2, a star configuration using the coupling facility is supported as an alternative.

D

DASD **direct-access storage device**[s], disk storage.

DAT dynamic address translation.

Data Control Block (DCB) MVS control block or its defining macro-instruction, the internal (to a program) representation of a data set.

data management portion of MVS responsible for handling input and output operations at the level above physical device management; the major portion of the Data Facility Product (DFP) now part of each MVS system.

data set IBM terminology for an externally manageable unit of data, often called a *file* in other systems. A data set is contextually defined as the entity described by a DSCB on DASD, as a physical file on magnetic tape, as the portion of the JCL data stream following a "`//SYSIN DD *`" or "`//SYSIN DD DATA`" statement, or as a SYSOUT data stream.

Data Set Control Block (DSCB) data area on a DASD describing the location, content, format, and size of a data area on that device. DSCBs exist in several formats, the most common being the Format 1 DSCB describing a non-VSAM data set. The DSCB is the generalization of the file label or directory entry of other systems.

data set organization (DSORG) physical format of a data set on its containing medium. DSORGs include *physical sequential*, *partitioned*, *direct*, and *VSAM*.

data set, linear type of VSAM data set used to back a Data in Virtual object or a scroll hiperspace.

data space MVS/ESA variant of an address space containing no common segments; therefore a full 2 gigabytes of private space. Data in a data space is addressable only in access register (AR) mode and cannot be executed as instructions.

data stream data organized as a continuous stream of bytes, given meaning as individual data fields by the program reading or writing it. The concept of *stream I/O* originated in FORTRAN and was further formalized in PL/I. The contrasting type is called *record I/O*, consisting of discrete records of fixed or variable size. I/O to and from a UNIX terminal is stream data as well.

data windowing services set of services provided in MVS/ESA to make hiperspaces and Data in Virtual objects accessible to programs written in high-level languages.

dataset non-IBM spelling of *data set*.

DB2 IBM's flagship relational data base subsystem.

DCB see **Data Control Block**.

DCC see **disconnected command chaining**.

default behavior of a system in the absence of specification or control information—what the system will do on its own.

default parameter values assigned to a control variable if no value is communicated to the system.

defragmentation process of consolidating and coalescing data sets on DASD to make each data set occupy a single contiguous extent. For data sets that can contain internally unused and unusable space ("gas"), "defragging" usually includes "degassing" as well.

degradation analysis performance monitoring technique that samples and breaks down the time taken by a process into its component parts, classifying each sample into the appropriate state, and accumulating data to produce a table of state frequencies. The data so collected may be displayed in various forms. Other names for degradation analysis include execution-state analysis, **bottleneck analysis**, and response-time component analysis.

delay state state found in degradation analysis in which the workload of interest is not receiving service because it is waiting for access to a resource that is not currently available [to it].

demand paging in SRM, the sum of nonswap, non-VIO page-ins and page-outs; in common parlance, page-ins caused by page faults. This is a case in which the SRM term is contradictory to common understanding.

detected wait a wait (without the LONG parameter) that exceeds a duration limit set by the SRM. Address spaces sustaining detected waits are candidates for swap-out. At this writing , the duration limit is 8 SRM seconds or 2 seconds, whichever is longer. In current systems likely to be running MVS, the 2-second limit is the operative one.

device utilization the average (mean) portion of a time interval during which the device in question was found to be busy—the target of a currently active I/O operation. Devices are either busy or not busy, but utilization can be any value from 0 to 100 percent.

DFP Data Facility Product; see **data management**.

DFSMS Data Facility Storage Management Subsystem; that part of MVS/ESA that implements system-managed storage. See **SMS** for more information.

DFSMShsm current name of IBM's Hierarchical Storage Manager, a data management extension that handles archiving, backup, and migration of data among levels of a storage hierarchy.

direct access storage device see **DASD**.

disconnect time time in the execution of a channel program in which disconnected command chaining is initiated.

disconnected command chaining (DCC) the capability of an I/O device (through its control unit) to break its connection with the channel, allowing an operation that does not require data transfer to proceed without tying up the channel. DCC allows block multiplexor channels to be used for multiple concurrent DASD operations, since the time such a device is disconnected usually far exceeds its connect time.

discretionary in Workload Manager goal mode, the set of work units classified as having no goal and no importance. Workload Manager has discretion to provide service to such work units as resources are available after all achievable goals are met. The service received by discretionary work units depends strongly on the availability of surplus resources. In real storage management, discretionary pages are those awaiting page-out, subject to the discretion of RSM as to its current need for the frames occupied by such pages.

disk the IBM way to spell the word for a thin, circular, spinning (usually magnetic-coated) object found in a DASD. Some non-IBMers and (especially) anti-IBMers make it an article of religious faith to spell the word *disc*.

DIV Data in Virtual.

DSCB see **Data Set Control Block**.

DSORG see **data set organization**.

DSRV IPS parameter specifying the minimum and maximum service rates corresponding to contention index values of 100 and 1, respectively, for a domain. The contention index is determined based on the total service rate of all address spaces in the domain.

dual processors computer configuration typified by larger models of the IBM 4381 in which two processors (CPUs) share common storage and each has a complement of attached cycle-stealing channels. Such a configuration is not physically partitionable and, in MVS/XA and MVS/ESA environments, does not intrinsically provide the independent channel subsystem of those architectures. The channel subsystem is simulated through microcode, but "channel set" considerations are still necessary in system planning. This design has been superseded by dyadics in the IBM ES/9000 9121 line.

dyadic computer configuration in which two processors share common storage and an independent channel subsystem. Such a configuration is not physically partitionable.

dynamic path reconnect (DPR) ability of a DASD subsystem (control unit or storage director, string controller, and device) to reestablish an I/O connection after completion of a disconnected operation through any available storage director to any available channel connected to the

initiating system. DPR is not supported in MVS/370, requiring the path-definition facilities of the XA or ESA channel subsystem.

dynamic pathing DASD subsystem facility consisting of two parts, dynamic path selection and dynamic path reconnect.

E

EBCDIC Extended Binary Coded Decimal Interchange Code, IBM's proprietary character encoding based on the old IBM punched-card code. EBCDIC was picked as the default code for OS/360 and became the only character set code for System/370 and System/390 as the "ASCII bit" was dropped from the system architecture. In practice, EBCDIC is not a great problem in interchanging data between the System/390 platform and other, ASCII-based platforms. Translation is simple and efficient. However, EBCDIC is a symbol of IBM's proprietary past and represents an objection to be overcome in some close choice situations.

element in OS/390, a part of the operating system (such as VTAM) that was formerly a separate product or product feature.

enabled attribute of a system function or subsystem capable of taking advantage of an advanced capability of MVS since Version 5. Workload Manager goal mode, parallel sysplex data sharing, and VTAM generic logon are examples of such capabilities.

enclave an independent unit of work whose priority may be different from the priority of the requester. The only IBM-provided requester to date is the distributed data facility (DDF) of DB2 Version 4 and later. An enclave is created using a WLM interface, providing classification information that is used to determine the service class for the unit of work. In compatibility mode, the service class is used to select a performance group. One or more SRBs are scheduled associated with the enclave. The SRB will be dispatched with addressability to some address space, such as the server which received the request. The SRB will be queued, dispatched and preempted based on the enclave's priority (from the current service class period or performance group period) rather than the priority of the address space that is known as its home space.

engineering embodiment physical realization of an architecture, usually defining a system or product-family of systems; it may not implement all of the architecture or the full extent of the quantitative limits of the architecture. For instance, XA defines 2 gigabytes as its maximum real (central storage); no model of the 308X product family had more than 128 megabytes.

enqueue lockout condition in which a serially reusable resource controlled by the ENQueue service is held by a work unit that will not let it go. A typical cause is when the holder of the exclusive ENQ in turn requires another resource held by yet another work unit. The root problem is often known as a *deadly embrace*, but that term normally refers only to the symmetrical situation. (There are exactly two affected work units, and each has what the other wants.) Other work units (usually each is a separate address space) are merely locked out.

ES Expanded Storage.

ESA Enterprise System[s] Architecture.

Ethernet a local area network (LAN) developed by Xerox, Digital and Intel (IEEE 802.3). It is the most widely used LAN access method. Ethernet connects up to 1,024 nodes at 10 Mbps over twisted pair, coax and optical fiber. Faster Ethernets have been developed. 100BaseT and 100VG- AnyLAN transmit at 100 Mbps rather than 10, and switched Ethernet gives each pair of users that want to communicate with each other a dedicated 10 Mbps channel.

Ethernet is a data link protocol and functions at the data link and physical levels of the OSI model (1 and 2).

exabyte 10^{18} bytes (in practice, the power of 2 value that just exceeds the decimal value, 2^{60}). *Exa-* is a relatively new metric prefix, and there are yet newer ones. Here is a list of currently defined prefixes; the powers of two are not metric values but are commonly used to denote memory sizes and sometimes personal computer hard disk sizes:

Prefix	Symbol	Magnitude	Meaning (multiply by)
Yotta-	Y	10^{24} or 2^{80}	1 000 000 000 000 000 000 000 000
Zetta-	Z	10^{21} or 2^{70}	1 000 000 000 000 000 000 000
Exa-	E	10^{18} or 2^{60}	1 000 000 000 000 000 000
Peta-	P	10^{15} or 2^{50}	1 000 000 000 000 000
Tera-	T	10^{12} or 2^{40}	1 000 000 000 000
Giga-	G	10^{9} or 2^{30}	1 000 000 000
Mega-	M	10^{6} or 2^{20}	1 000 000
Kilo-	K	10^{3} or 2^{10}	1 000
Hecto-	H	10^{2}	100
Deka-	D	10	10
deci-	d	10^{-1}	0.1
centi-	c	10^{-2}	0.01

Prefix	Symbol	Magnitude	Meaning (multiply by)
milli-	m	10^{-3}	0.001
micro-	μ	10^{-6}	0.000 001
nano-	n	10^{-9}	0.000 000 001
pico-	p	10^{-12}	0.000 000 000 001
femto-	f	10^{-15}	0.000 000 000 000 001
atto-	a	10^{-18}	0.000 000 000 000 000 001
zepto-	z	10^{-21}	0.000 000 000 000 000 000 001
yocto-	y	10^{-24}	0.000 000 000 000 000 000 000 001

exception analysis monitoring technique in which states or quantities observable within a system are continually compared with acceptable states or values. When a deviation is found, an *exception* is said to exist. The monitor may display a message describing the exception on a terminal associated with it or on the system console, store the exception data on a data set, send it to SYSOUT, or take any of several other possible actions. Depending on the type of data examined, the monitor may be performing system availability analysis, service delivery analysis, response time analysis, elapsed time analysis, and so on.

execution-state analysis see **degradation analysis**.

expert system branch of artificial intelligence known also as a rules-based system. Elements include a *knowledge base*, describing the rules by which decisions are to be made, and an *inference engine*, a program that will follow the rules in any knowledge base to yield unambiguous conclusions. Expert systems are usually driven in a *consultation* by a human user; some expert systems gather data from performance monitors to make recommendations aimed at improving system performance.

extent contiguous portion of a DASD containing all or part of a data set. A data set with multiple widely distributed extents tends to be inefficient in random-access performance.

externalize to make a portion of a system subject to change or control without requiring knowledge of internal structure. MVS's OPT parameters prior to MVS/SE2 were limited, and system changes were often made by changing the values in internal control blocks. Many of those values were externalized—became OPT parameters in MVS/SE2 and subsequent releases, thus reducing the exposure to MVS integrity.

externals parts of a system subject to control by parameters or commands; also the descriptions of such parameters and commands.

F

fencing early name for storage isolation.

fork in UNIX or OpenEdition MVS, a copy of a process created for the purpose of independent excution, or the act of creating the copy.

fragmentation see **defragmentation**.

frame the unit of real storage, 4096 contiguous bytes beginning at a real storage address divisible by 4096, the same size as a page.

front-end the portion of I/O processing preceding the issuing of the instruction initiating the I/O; see **back-end**.

G

GETMAIN MVS service routine and its invoking SVC; allocates virtual storage upon program request. In MVS/ESA, GETMAIN (as well as its companion deallocation service FREEMAIN) has been superseded by a more comprehensive STORAGE service, rather than expanding GETMAIN to deal with allocation of storage in data spaces.

granularity in capacity planning, increments of size or cost in a feature such as memory or CPU capacity. The smaller the increments, the higher the granularity.

GRS Global Resource Serialization.

GTF Generalized Trace Facility.

GUIDE an association of users of IBM computers. GUIDE, and a similar association called SHARE, contribute to the development and refinement of IBM hardware and software through technical conferences and exchanges, as well as a formal requirements process.

H

happy values obsolescent name for SRM's MPL adjustment parameters, from the notion that the SRM is "happy" and therefore not about to change the MPL if each measured value is within its specified range.

hashing algorithm

data structure algorithm that distributes data according to some function of its key value so that searching the structure approximates ideal validity of reference. Simple hashing algorithms (such as assigning each letter of the alphabet to its own area) lead to numerous collisions and the need for supplementary local tables; more complex algorithms may require very large virtual filing cabinets.

head of string control point of a DASD subsystem usually contained in the "A box" of the string. (In DASD subsystems with optional modules, the A box is the first or only required component. The name derives from IBM's customary model designation.) Since the string controller is part of a DASD path, it represents a site of contention and queueing, usually called "head-of-string delay." In some past devices, the actuators physically present in the A box may have had slightly less protocol delay and some priority in processing concurrent requests. The differences today are negligible at most.

hardware compression compression implemented in the CPU or channel subsytem of the processor complex, or in an I/O device controller.

hierarchical file system (HFS) In general, a file organization method that stores data in a top-to-bottom organization structure. All internal access to the data starts at the top and proceeds throughout the levels of the hierarchy. Most operating systems use hierarchical file systems to store data and programs, including DOS, OS/2, Windows NT and 95, UNIX and the Macintosh.

In particular, the facility implemented in DFSMS to provide OpenEdition MVS with ability to read and write files in a UNIX HFS.

high-speed buffer (HSB) fast storage associated with the instruction fetch and processing elements of a CPU: a cache. For example, a CPU with a 15-nanosecond cycle time would have to take three idle cycles while waiting for data or instructions from a 60-nanosecond memory element. If the data and instructions are asynchronously staged into a 15-nanosecond HSB, CPU operations could proceed without the inserted delays. Correctly sizing an HSB is a classical exercise in cost/performance analysis.

HSM see **DFSMShsm**.

hypervisor a control program whose only function is to facilitate the running of other control programs or operating systems. IBM's PR/SM is a hypervisor embodied in hardware and microcode. VM is a somewhat compromised hypervisor because it supports numerous functions that allow virtual machines to communicate with it, in recognition of practical performance realities. A "pure" hypervisor insulates its various subordinate operating environments from one another; to the extent that it approximates an operating system, that benefit is diluted.

I

ICS Installation Control Specification, the IEAICSxx member of SYS1.PARMLIB, used to assign address spaces to performance groups in compatibility mode.

IDAW Indirect Data Address Word, a logical extension to a CCW that specifies data address and count; it may contain a 31-bit address and was the means of enabling MVS/370 to support I/O to real storage addresses exceeding 16 megabytes.

image processing form of data processing in which digitized images of actual documents are stored and manipulated in order to save transcription time and avoid transcription error. Large potential cost savings are offset by very high computer resource requirements. The image of a typewritten page may require 50,000 bytes to store; as text it may take 3000 bytes.

IMS IBM's Information Management System, a high-function subsystem providing both data base and data communication (transaction management) functions. IMS was one of the first such subsystems, dating back to OS/360 and continually updated to exploit MVS advances including parallel sysplex and Workload Manager goal mode. The current version is known as IMS/ESA.

Indirect Data Address Word see **IDAW**.

Initialization and Tuning IBM publication, in a different edition for each version of MVS, that specifies (or points to) the operating parameter choices for an MVS system. Since Version 4 of MVS/SP, there have been two separate publications, *Initialization and Tuning Guide* and *Initialization and Tuning Reference*. The former describes algorithms and concepts; the latter describes parameters and their permissible values.

Installation Control Specification see **ICS**.

Installation Performance Specification see **IPS**.

IOSQ That portion of an I/O operation spent waiting while queued on a busy **UCB.**

IPL Initial Program Load; hardware-defined operation leading to the [re]initialization of a System/390 operating system including MVS.

IPS Installation Performance Specification, a member of SYS1.PARMLIB that contains parameters directing the SRM in its assignment of resources to workloads in compatibility mode.

ISDN Integrated Services Digital Network, a telephone network based on digital signals, capable of data rates of up to 128Kbits per second.

ISPF Interactive System Productivity Facility, full-screen text interface to TSO and a great contributor to TSO's acceptance.

J

Java a platform-independent programming language developed by Sun Microsystems, based on C++. The most common use of Java is to create small applications (**applets**) which are made available for download from World Wide Web sites on the Internet. The applets are executed on the receiving computer in a Java Virtual Machine environment, usually supplied by the software used for browsing the Internet.

JCL job control language.

JES job entry subsystem. In response to MVT's deficiencies at managing job sequencing, printers, and input data streams, two rival subsystems were designed and implemented primarily by customers with IBM support at only the local level. Both were eventually acquired by IBM. The Houston Automatic SPOOLing Priority system (HASP) eventually was made into JES2; the Attached Support Processor (ASP) became JES3. As the two JESs converge in function and performance, they are more and more distinguished primarily by their mutually incompatible control languages. The time may yet come when there is only one JES, but it probably won't be soon.

JES2 JES2 started out as the smaller, faster, simpler JES. It has since grown in functionality so that most of what JES3 can do is found now in JES2, perhaps with extensions built on its numerous exits. See also **JES**.

JES3 JES3 began as the high-function JES for multisystem environments and those with complex batch jobstreams requiring dependency scheduling. JES3 performance now rivals that of JES2. See also **JES**.

job control language (JCL) Control statements that define a batch job, started task, or subsystem, existing in 80-byte fixed-length records or *card images*.

job entry subsystem see **JES**.

jobstream sequence of batch jobs. Usually the term refers to a related sequence or *suite* of jobs that accomplish a single business purpose. For example, a General Ledger jobstream might consist of 30 or more individual jobs with sequential dependencies to be controlled by a production control staff or an automated job management system.

K

K in scientific and engineering notation, the quantity 1000; in binary-based computing systems, the quantity 2^{10}, or 1024.

L

late binding see **binding**.

latency the mean rotational delay of data located at any angular position on a DASD track; generally estimated as one-half of the rotational period.

linear searching simplest data searching technique, in which the element to be matched is compared with the corresponding part of each stored data structure. If the list is ordered, a miss requires an average search of one-half of the data extent; for an unordered list, a miss requires a full element-by-element search, the worst-case violation of validity of reference. Linear searching requires no setup; therefore, if the setup time of a more efficient search method is greater than the time needed to traverse the list (or half if ordered), linear searching may be an appropriate choice. The bad reputation of linear searching comes from its fixed time requirement per element; more efficient methods have a fixed setup time and a variable time component that is less than proportional to the number of elements. Once the crossover point (the number of elements at which competing methods have equal search times) is passed, linear searching becomes increasingly costly in both execution time and working-set size.

LLA in MVS/XA, *LINKLIST Lookaside Area*, an address space that contains the directories of LINKLIST libraries, used to avoid outboard searching for load modules. In MVS/ESA, *Library Lookaside Area*, a service employing data spaces through VLF services to contain directories and load modules of selected LINKLIST libraries, used to avoid both searching and Program Fetch I/O.

locality of reference design principle and common property of programs; instructions and the data they use tend to be tightly clustered, whether the data is in virtual storage or on DASD. The clustering is both spatial and temporal; temporal clustering is sometimes called *temporality of reference*. If the implementation environment does not enforce locality of reference, it may be necessary to enforce it as a design principle, to ensure minimum working-set size for internal data and minimum access delay for external data.

"loved one" a workload that is economically significant to the business unit operating a data center; therefore it is closely watched and rescued if it is headed for trouble.

LPAR Logical PARtition in a PR/SM or MLPF environment.

LRU least recently used, page-stealing strategy which assumes that a page not recently referenced is not likely to be referenced again soon. For programs exhibiting strong locality of reference, this is usually a

sound assumption. There are, however, many instances in which LRU is counterproductive. If a transaction-processing subsystem serves a large population with little commonality in the transactions they invoke, the correct strategy may be to discard the pages of a transaction as soon as it is completed. This "MRU" (Most Recently Used) strategy resembles swapping more than LRU page stealing. The original studies that resulted in LRU's becoming the standard page replacement strategy may need to be reexamined with today's workloads.

M

M in scientific and engineering notation, the quantity 1,000,000; in binary-based computing systems, such as the ones that support MVS, the quantity 2^{20}, or 1,048,576.

M/M/1 queueing model which assumes an exponentially distributed arrival rate (first "M"), exponentially distributed range of service times (second "M"), and a single server ("1"). M/M/1 is a useful model because it is mathematically simple and does match well to some common delay scenarios in data processing. As with any model, the scenario under consideration should be carefully checked against the assumptions of the model before using it.

main storage old name for storage internal to a computing system, as opposed to DASD or tape. The term evolved to *real storage* when virtual storage came in, then to *processor storage* in preparation for the announcement of expanded storage, and finally to *central storage* when expanded storage was announced.

mainframe a large computer system, particularly one on which an operating system like OS/360 or MVS runs. The term originally referred to the contents of the large cabinet in which the processing unit (CPU) resided. The hierarchy (since the first integrated circuits) was chip, module, board, gate, frame. The main or most important frame was the one containing the CPU and its local storage.

maximum PWSS storage-isolation parameter designating the maximum value an address space's target protected working set may reach. This is also the central storage frame count above which the address space is subject to preferred page stealing. A limited maximum PWSS should be specified *only* when preferred page stealing is desired.

MC MONITOR CALL instruction.

MDF Multiple Domain Facility, Amdahl Corporation's name for a hardware hypervisor providing multiple system images on a single processor complex. It preceded IBM's PR/SM in the marketplace.

mean-time-to-wait (MTTW) the average interval between invocations of the *WAIT* macro or service by an address space, usually for I/O operations. MTTW is used explicitly in compatibility mode as a dispatching priority specification. The M*n* specification denotes a range of ten dispatching priorities. Address spaces with that specification (*e.g.* **DP=M3**) are observed by SRM to determine the MTTW of each; their individual dispatching priorities are set depending on those values. The top priority in the group of ten goes to address spaces with MTTW of 0-45 SRM milliseconds, with each additional 45 SRM ms dropping the priority one notch. The step size may be changed from the default of 45 SRM ms by changing the value of the OPT parameter CCCSIGUR.

In goal mode all work units classified as discretionary are in a single MTTW dispatching group. CCCSIGUR is ignored in goal mode; presumably the spreading of priorities is done on a more heuristic basis.

megabyte see **M**; 1 M bytes.

MFT Multiprograming with a Fixed number of Tasks, the simpler implementation of multiprogramming in OS/360. Storage was divided into fixed-size partitions, as opposed to the more flexible regions of **MVT**.

microcode a level of programmed operation beneath the architecturally defined "Principles of Operation" interface. In most of today's systems, hardware provides a low-level machine not generally seen by the operating system; the power-on reset process initiates loading (from an internal storage device) and execution of microcode to initialize the hardware to be, for instance, a ES/9000-982 in LPAR mode. At some other activation, one side of the 982 could be the equivalent of a 941 in ESA/390 mode, the other a 941 as a coupling facility. Microcode programming (microcoding) is usually considered to be an engineering discipline, but the required skills are exactly those of a very flexible assembly language programmer. Underlying engineering designs may have seemingly arbitrary word sizes, and "instructions" that perform complex compound operations. Microcode also provides support for extended operation codes available to operating system (supervisor state) code to perform functions such as creating an MVS address space.

migration movement of data from one medium (usually low in capacity, fast, and expensive) to another (usually bigger, slower, and cheaper). Migration in a paging subsystem is used to move long-unreferenced pages from a device high in a storage hierarchy to one at a lower position. It was first used in VM's predecessor, CP/67, in the days of IBM 2305 fixed-head files (drums), moving pages from the few slots on the expensive drums to relatively unlimited cheaper storage on DASD. In MVS/XA and MVS/ESA, migration moves pages from

expanded storage to auxiliary storage. Just as in the CP or VM case, expanded storage migration requires frames in central storage as I/O destinations for incoming pages and origins for outgoing pages. Migration is triggered by internally maintained thresholds of available frames in expanded storage. Since MVS/ESA SP 4.2, the thresholds are "externalized" in the **OPT**.

migration age measure of constraint in expanded storage. The migration age is a property of the entire expanded storage resource. Low migration ages are usually correlated with high contention for expanded storage pages. The migration routine, or *migrator*, operates on demand when the available expanded storage frame count falls below a threshold and continues operating until the count passes a higher threshold. Each subsequent invocation of the migrator takes up where the previous one left off, returning to the first frame when the last one has been examined. The migrator examines the control bits of each frame in sequence, turning off the "new" indication at first encounter and selecting non-new frames for migration provided they are not in a protected class, such as a storage-isolated working set.

Migration age is the time required for the migrator to traverse the entire set of expanded storage frames and return to its starting point. It is a complex function of the size of expanded storage and the workload mix on the system. It is useful to measure and track migration rate as a function of migration age. The "knee in the curve" defines a *critical migration age* that may be used as the lower limit of criteria age for less-important workloads.

MIPS millions of instructions per second, a measure of CPU power. However, not all instructions are alike, and IBM in particular does not generally release MIPS data for its products. Bare measures like MIPS tend to ignore important CPU performance factors, including storage and I/O bandwidth, microcoded assists for operating systems and key programs, concurrency, and the power of individual instructions.

MLPF Multiple Logical Processor Facility, Hitachi Data Systems' name for a hardware hypervisor providing multiple system images on a single processor complex. MLPF is similar in functional capability to IBM's PR/SM.

MP multiprocessor.

MPL multiprogramming level, the number of address spaces resident in central storage at any given time; also the number of address spaces of a particular kind or in a particular domain.

MPL delay delay state in which an address space is swapped out and ready to be swapped in, but SRM cannot increase the MPL in its domain to allow the address space to be swapped in. MPL delay is also

known as "out and ready" time. The Workload Manager in goal mode will place address spaces in this state as one of its mechanisms to regulate performance to goals.

MSO (1) component of service unit accumulation based on Main Storage Occupancy; (2) service definition coefficient (SDC) in the IPS or Service Definition establishing the weight of MSO service units in the overall service unit quantity.

MTTW see **mean-time-to-wait**.

Multiple Domain Facility see **MDF**.

multiprocessor a computer hardware configuration in which two or more CPUs or processors operate under control of a single instance of a control program or operating system. In System/390 multiprocessors, all processors have access to all storage and channels. One-sided configurations can have up to five processors in the IBM ES/9000 711 series and in the Multiprise 2000, and up to ten processors in the IBM 9672. Two-sided configurations such as the larger ES/9000 9021 models are always multiprocessors; they have the added potential to be physically partitioned into two independent hardware configurations.

multiprogramming method of operating a computing system so that multiple workloads are *concurrently* active. In addition, if the system is a multiprocessor, as many workloads may be *simultaneously* active as there are processors. When one workload gives up its use of the CPU by entering the WAIT state, another workload can be dispatched. In earlier times, batch multiprogramming exploited I/O waits; in current systems, multiprogramming exploits the "think time" of terminal users as well.

multiprogramming level see **MPL**.

multiprogramming set the set of address spaces that constitutes the system-wide MPL.

multitasking the possibility of multiprogramming within a single address space. The ATTACH SVC creates an additional dispatchable unit (task), quasi-independent of the task that created it. The **FORK** function in UNIX or OpenEdition MVS performs a similar function although the "child" may be a separate address space.

MVM original name for MVS, Multiple Virtual Memories.

MVT most complete version of OS/360, Multiprogramming with a Variable number of Tasks. In MVT, storage was divided into *regions* of variable size.

N

negative storage isolation use of storage isolation with a low maximum PWSS to designate an address space as a preferred source for page stealing. Although it is effective, negative storage isolation conflicts with the Working Set Manager and should not be used for swappable workloads in MVS/ESA SP 4.2 and later systems.

network operating system (NOS) an operating system that controls a server system, with extensions for communication with many client systems as well as functions for making shared devices available to clients. Novell NetWare, Windows NT Server, and OS/2 Server are popular examples. OS/390 has all the attributes of an NOS as well.

network-centric computing IBM's term, as expressed by its Chairman, Lou Gerstner, for a computing environment in which major applications and their data are maintained on one or more large server systems (presumably OS/390). Clients connect to the server(s) through the Internet or local networks over high-speed connections, downloading programs and data as needed. The development of the **Java** language by Sun Microsystems has been enthusiastically endorsed by IBM as a direction-setter for network-centric computing.

nonparametric statistics method of statistical analysis that does not assume a particular model for the distribution of data. Statistics that depend on the ranking of data rather than on a distribution model are characteristic of nonparametric statistic. Examples include the median, quartiles, and percentiles.

nonswappable address space not eligible for swap-out. Many started tasks that run in authorized state simply make themselves nonswappable by issuing the appropriate SYSEVENT SVC. Others become nonswappable by means of an entry in the Program Properties Table, and any address space operating in cross-memory mode is nonswappable by definition.

normal distribution a statistical model (also known as Gaussian) that is completely characterized by two random variables, μ (mu), the mean, and σ^2 (sigma-squared), the variance. A plot of the normal distribution is the familiar bell-shaped curve. A full discussion of normal random variables may be found in Arnold Allen's *Probability, Statistics, and Queueing Theory With Computer Science Applications* (Academic Press,1978).

The normal distribution is tempting to use because it is very easy to calculate a "mean" and "standard deviation" (to a first approximation the square root of the variance) from a series of observations. However the calculated standard deviation will be misleading and inappropri-

ate to use if the data do not conform to the normal model. The transaction completion times shown in an RMF Workload Activity Report often show standard deviations exceeding the magnitude of the means—simply because a population of transactions is very unlikely to meet the assumptions of the normal distribution model.

nucleus the permanently resident portion of MVS, encompassing the I/O supervisor, the various resource managers (ASM, RSM, SRM, etc.), interrupt handlers, and the most essential service routines. The "horizontal splitting" made possible by cross-memory services complicates the definition. ENQ/DEQ services used to be in the nucleus, but now these services are provided by GRS in a separate address space.

numerically intensive [computing] (NIC) computer application commonly encountered in scientific and engineering work, characterized by very long intervals between WAITs and thus near-continuous demand on the CPU resource. In many such applications, there is also very heavy demand on virtual storage, with a correspondingly large use of central storage. The working set is large and volatile, making such a workload difficult to manage in a system with even the smallest amount of central storage constraint.

NIC work was ill-served in MVS systems prior to the introduction of Working Set Management in MVS/ESA SP 4.2. It was typically scheduled to run at otherwise idle times because the paging activity engendered by the large and constantly-changing working set disrupted the stability of other work. When NIC work is also important work, it cannot be pushed aside. Working Set Management enhanced the SRM to observe the working set behavior of address spaces and the amount of CPU devoted to the paging task, and to take action such as bulk page stealing and swapping to maintain an optimal mix of work in central storage. Workload Manager in goal mode uses the mechanisms of Working Set Management as one of its tools to manage workloads to meet goals.

O

open system this term has had many meanings. At one time, it was a code term for "UNIX" in contrast with the perception that IBM's systems were proprietary and therefore closed. As MVS has evolved into OS/390 and has become certified for the highest level of UNIX compliance, it is now an open system in the most polarized meaning of the term. Currently the "open" designation is applied to systems that can communicate with, and inter-operate with a wide variety of systems and platforms using standard protocols and interfaces.

operator person responsible for directing the activities of a computing system. Key operator tasks include [initiating] system startup and shutdown, observing and replying to messages displayed at the system console, entering commands to deal with unusual circumstances, mounting and demounting tapes, and loading and removing printer paper. All of these tasks except those dealing with printer paper are candidates for automation. Automated tape libraries and console automation programs are redefining the operator's job away from performing mundane tasks and toward high-level decision making. Paper handling is becoming less important as more high-quality page printers are installed at the point of need rather than in the central computer room.

OPT abbreviation for "optimizer," an early name for the SRM. Also, usual designation for the IEAOPTxx member of SYS1.PARMLIB.

out-and-ready condition of an address space that is swapped out and ready to run, but cannot be swapped in because the MPL in its domain will not allow a unilateral swap-in and there is no candidate in the domain eligible for an exchange swap.

overcommitment use of a resource beyond its physical limits. The term is usually applied when virtual storage is allocated in greater aggregate quantity than real (central) storage. A form of CPU overcommitment is recognized by the MVS SRM when not all members of the multiprogramming set receive CPU service during a measurement interval.

overinitiation pejorative name for the condition in which the aggregate MPL of batch jobs is less than the number of active batch jobs. In fact, overinitiation is a prerequisite to effective MPL management.

overlapped processing characteristic of a computing system in which I/O operations proceed concurrently with CPU processing. Independently operating channels make overlapped processing possible. Overlapped processing in turn makes multiprogramming and multitasking possible.

override to substitute a selected value for the default value of a parameter.

P

P/390 a heterogeneous multiprocessor consisting of an IBM PC Server using the microchannel architecture, with an add-in card (also called P/390) that supports the full ESA/390 instruction set with the exception of the coupling facility instructions. The combined system is built on OS/2 and emulates System/390 devices through the underlying hardware, as well as accommodating external device attachment.

The P/390 in its larger configurations is a capable MVS platform, able to replace obsolete System/370 platforms like the 4341 and the 3083 for production use at smaller sites. It also is a very accessible development and test platform for applications; one suggested use is testing for Year 2000 preparedness. OS/390 can be ordered in a simple-to-install CD-ROM configuration for the P/390.

There is also an R/390, differing from the P/390 in that it is built on the RISC 6000 platform with AIX/6000 as the supporting operating system.

page *n.* 1. the unit of virtual storage, 4096 contiguous bytes at a starting address divisible by 4096; 2. the unit of expanded storage, 4096 bytes designated by a single page number; *v.* to move pages between slots of auxiliary storage and frames of central storage; usually the movement from auxiliary to central (page-in) is of interest.

page fault an interrupt caused by a page-translation exception, signifying that the page is not shown to be occupying a frame of central storage. A page fault results in a page-in, usually of a block of related pages.

page movement the transfer of pages between central and expanded storage; also the movement of frames from central storage above 16 megabytes to and from addresses "below the line," used in MVS/370 to allow ordinary I/O (without IDAWs) to proceed.

page stealing activity initiated when the Real Storage Manager (RSM) has allocated a frame and finds that the available frame count has fallen below the current MCCAFCLO threshold. The condition is also triggered in conjunction with a swap-in when the required number of frames would put AFC below MCCAFCLO. SRM selects the frames to be reassigned, first taking frames from storage-isolated address spaces with frame counts above their PWSS maximums, then taking frames from storage-isolated address spaces with more frames than their current targets as well as other address spaces and the common area, in descending order of UIC. SRM sends any required page-outs to the auxiliary storage manager (ASM).

parallel sysplex see **sysplex**

parallel-coupled (author's term) the form of multiprocessing architecture embodied in the parallel sysplex. The essential elements are:

- multiple MVS images, each with its own independent copy of the base control program and all subsystems, at release levels that support the coupling facility and data sharing
- connections from each MVS image to and from one or more coupling facilities

- sharing of application data and some system data through ESCON channels and directors

PARMLIB partitioned data set (SYS1.PARMLIB) containing the externally specifiable parameters of MVS and some of its subsystems. *Initialization and Tuning* contains a description of each member of SYS1.PARMLIB.

partitioned data set (PDS) data set organization (DSORG) consisting of a directory and a set of members resembling individual sequential data sets. A PDS is often known as a library.

partitioning, physical the act of dividing a computer system into two (or more) independent computer systems. Physical partitioning requires separate components from power supplies on up. IBM systems and compatibles capable of running MVS support (to date only) two-way physical partitioning if appropriately configured in "two-sided" configurations.

path an I/O connection between a CPU and a device consisting of a channel, control unit, and device controller, if applicable, as well as the device.

PEND in XA and ESA, the time spent by the channel subsystem waiting for a path to become available. PEND time is often associated with shared DASD contention; it represents the I/O activity on other systems that prevents an operation on the current system from proceeding.

performance index (PI) a normalized metric used by the Workload Manager in goal mode, representing the degree to which each specified goal is being met. It is defined for all but velocity goals as (actual measurement) ÷ goal; for velocity goals it is the reciprocal. A goal just met gives a PI of 1; a goal missed has a PI greater than 1, and an overachieved goal has a PI less than 1.

petabyte 10^{15} bytes, sometimes the nearest binary equivalent, 2^{50}. See **exabyte** for a table of metric scale prefixes.

physical sequential data set organization (DSORG) in which logical records (defined by use in a program) are organized in (often larger) physical records (defined to optimize device performance) and read or written in strict order from the beginning to the end of the data set.

physical swap movement of the trimmed working set of an address space between central (real) storage and auxiliary storage. Although a processor storage swap is also a physical phenomenon, the term is used only for the swap involving the use of auxiliary storage.

PLPA pageable link pack area, part of the common area, also including EPLPA in MVS/XA and MVS/ESA, containing **reenterable** load modules that may be used (in place) by any address space. There is

also a fixed link pack area, containing modules that may not sustain page faults and therefore are ineligible for page stealing.

PLPA page data set the first-named page data set in an MVS IPL, containing only pages of the PLPA. Since the PLPA is read-only, the PLPA page data set is preserved across IPLs and is rewritten only when the CLPA option is specified for the IPL.

portable application application program or system written so it can be executed on several different kinds of computer systems. In general, portable applications trade off ease of implementation in environments other than the first for less-than-optimal performance in each environment. Of necessity, they must use a "lowest common denominator" of system facilities and services in order to be portable. Implementors or vendors of such applications must often devote substantial expense to improving performance, reliability, integrity, and flexibility in particular environments, in response to customer requirements.

PPT Program Properties Table, an MVS data area specifying names of programs that are to enjoy various forms of privilege.

preferred page stealing page stealing incurred by storage-isolated address spaces holding more frames than the maximum PWSS for the current performance period. See also **negative storage isolation**. Preferred page stealing also takes place in Workload Manager goal mode for an address space that has been identified as a donor of central storage frames.

Principles of Operation IBM publication defining hardware data structures and organization, instruction set, and related interpretations and relationships for a computer system or family of systems.

process the UNIX equivalent of an MVS task.

processor complex the central elements of a computing system, including processor(s), processor storage, and the inner portion of the I/O subsystem (in System/390 the channel subsystem).

processor storage central storage and expanded storage; all storage accessible to a processor.

Program Fetch service routine in the MVS nucleus that loads programs from libraries into virtual storage. In ESA, Fetch also loads programs from the LLA data space. Program loading is more complex than simply reading in a module; programs are generally relocatable, so internal references to addresses must be adjusted relative to the loading address of the module. The principal SVCs serviced by Fetch are those for ATTACH, LINK, LOAD, and XCTL.

Program Properties Table (PPT) see **PPT**.

program status word (PSW) in System/360, the 64-bit (double-word) representation of the current state of the CPU. In System/370, XA, and ESA, the PSW still contains the current instruction address and various mode and mask settings, but to capture the full status of the CPU requires the content of several control register fields as well. The PSW does not exist as a viewable object except when stored as the result of an interrupt. When the interrupt occurs, the PSW is created and stored at one of six reserved addresses according to the interrupt type, and a new PSW is fetched from a corresponding reserved address. A PSW at any double-word address is made the current PSW when it is the object of the LPSW (LOAD PROGRAM STATUS WORD) privileged instruction.

proprietary system opposite of an open system, a system in which data formats and protocols do not conform to industry standards.

PSW see **program status word**.

push-out the process of moving the trimmed working set of an address space from central storage to expanded storage, thus completing a processor storage swap-out.

push-out queue set of pages destined to be moved to expanded storage in FIFO order, as central storage is needed. The queue includes single pages (as from requested page-outs) and groups of pages (stolen blocks, VIO windows, trims, and swap groups), which are moved as groups.

Q

QSAM, queued sequential access method access method for physically sequential data sets or members of PDSs which accepts (PUTs) or provides (GETs) logical records one at a time. QSAM handles synchronization, blocking, and buffering without the need for problem program direction.

quiesce series of actions to prepare an address space for swapping; the principal task is to ensure that all pending I/O is completed. As of MVS/ESA SP Version 5, there is a **QUIESCE** operator command that swaps out a swappable address space and makes a non-swappable address space non-dispatchable.

R

R/390 see **P/390**

RAID (From an Internet citation entitled *RAID White Paper*):

In December, 1987, a University of California at Berkeley paper entitled "A Case for Redundant Arrays of Inexpensive Disks (RAID)" was

published by David Patterson, Garth Gibson and Randy Katz. The paper... noted that as the number of drives in a stripe set increases, the aggregate mean time between failure (MTBF) of the stripe set drops dramatically. The conclusion is that performance significantly improves at the expense of availability.

To solve the MTBF problem, the RAID paper introduced the concept of using redundant arrays to ensure data availability. The Berkeley paper describes five possible RAID methods, defined as RAID levels 1 through 5.

By definition, RAID has three attributes:

- It is a set of disk drives viewed by the user as one or more logical drives.
- Data is distributed across the set of drives in a defined manner.
- Redundant capacity or data reconstruction capability is added to recover data in the event of a disk failure.

Each RAID level explores these attributes in a different way.

Of the five RAID levels, RAID 1, RAID 3 and RAID 5 have received a significant amount of interest by the computer industry. As RAID has evolved into commercial products, the third letter of the acronym has come to mean "Independent."

RAMAC Name of the first commercially available disk storage product, the IBM RAMAC 305, introduced in 1956. In recent years the name has been revived for a family of new IBM devices based on RAID 5 arrays of small disks. The capacity of the original new RAMAC has been doubled in RAMAC II and doubled again in RAMAC 3. RAMAC 3 includes all the functions of the 3390-3990 combination and surpasses it in performance, capacity, reliability, and cost of ownership.

real frame replacement see **page stealing**.

reclaim see **page fault**.

reconnect the activity in the progress of an I/O operation following the successful completion of a disconnected command or series of commands; the device controller and storage director reestablish connection to the CPU through a channel. In System/370, reconnect could occur only on the original channel through which the operation began; in XA and ESA, any channel physically connected to the CPU and storage director or control unit may be used.

reenterability property of a program in which multiple concurrent activations may proceed without interference. In systems such as MVS, based on OS/360, reenterability was easy to achieve because programs generally acquire storage from the operating system (via the

GETMAIN SVC) for work areas and I/O buffers, and do not carry around such data in load modules.

reenterable attribute of a program that possesses the property of reenterability. The attribute is indicated by a bit in a load module's PDS directory entry.

reentrant [*sic*] common alternative designation for *reenterable*. This word is etymologically incorrect, having the plain meaning in other contexts of an entity that reenters itself. In a program, this would be a loop.

region in OS/360 MVT, a contiguous portion of real storage in which a program executed. The size of the region was a *de facto* limit on the aggregate size of all GETMAINs of the program in the region and all of its subordinate programs. In SVS, the storage became virtual, but *region* still functioned the same way. In MVS, the region became an address space of 16 megabytes, and later 2 gigabytes in XA and ESA. Region size would appear to have no current meaning. However, the page tables needed to map a full address space are sufficiently large that it is desirable to limit their extent if the full potential is not needed. Hence an MVS parameter and the **REGION=** specification in JCL or at LOGON allows a limit to be stated, separately in XA and ESA for both storage below 16 MB and above the line. The REGION limit in MVS retains the consequence of the MVT definition. It is the limit on the amount of virtual storage obtainable in an address space.

re-reference subsequent reference to data that was previously used. Re-reference to pages is what keeps their UICs low and thus protects them from stealing. Re-reference to blocks or tracks in a cache control unit prevents that data from being destaged to the backing device. In active TSO populations with some degree of storage constraint, re-reference to pages trimmed in the last few swaps accounts for about 60 percent of page faults. The use of storage isolation for TSO in such a case reduces TSO page faults dramatically at the low cost of moving a few more pages in the swap group. Re-reference frequency is the measure of locality of reference.

report class in Workload Manager goal mode, an optional entity created during workload classification that is used only to provide an alternative input to the RMF or equivalent reporting process. Each classified work unit may belong to no more than one report class. The data available for reporting is a subset of the data for a service class. The principal exception is that all service data is reported in a single performance period.

resource-oriented point of view approach to performance management that focuses on quantities measurable for hardware and op-

erating system resources, such as CPU and channel utilization, paging rates, and queue lengths.

RISC reduced instruction-set computer, a computer architecture optimized for high performance by restricting the number of instructions, registers, and instruction formats. In most RISC processors, almost all instructions may be executed in a single clock cycle, with some being overlapped so that more than one completes in a cycle. The contrast between RISC and **CISC** is diminishing as larger-scale integration allows both rich instruction sets and pipelined and parallel operation in evolutionary designs.

RISC designs are often different enough from other processors to require completely new software support, especially if the full performance potential of the design is to be attained.

RM1 interval at which SRM collects data, the SRM second.

RM2 interval at which the SRM in compatibility mode summarizes the data collected at RM1 intervals and decides whether to adjust the system's MPL. In MVS/370, RM2 was fixed at 20 seconds, but in MVS/XA and MVS/ESA it becomes smaller as the power of the processor complex increases.

RMF Resource Measurement Facility. RMF is described in Chapter 9 and discussed in Chapter 10.

RSM Real Storage Manager.

rule of thumb (ROT) method of making performance management decisions, based on comparing resource values with a set of values deemed to be acceptable. Deviations of the measurable values from the ROTs is taken to signify a performance deviation, irrespective of workload indicators such as response time. This method of performance management may sound absurd, but it is all too close to what MVS in compatibility mode does for itself by default.

S

scale-up problem performance problem that comes about when a workload volume increase makes formerly acceptable performance unacceptable. In particular, an algorithm that was efficient for low volume could become noticeably inefficient for high volume.

SCP System Control Program, one of two designations introduced on June 23, 1969, when IBM "unbundled" software and services from hardware, thus creating the mainframe software business. Originally, SCPs were provided at no cost with the hardware, and Program Products were available at a fee. As MVS evolved through MVS/SE to

MVS/SP, this distinction was lost, and today's counterparts to SCPs are not free.

sequential data set see **physical sequential**.

Service Level Agreement ideally, a carefully negotiated agreement between a data center and its customers committing the delivery of a specific minimum level of service (availability and performance) at a named maximum level of demand. In practice, it often turns out to be the expectation of the customers based on experienced or reported service attainment.

SHARE see **GUIDE**.

shared pages a facility allowing a page frame to be mapped to different private area addresses in different address spaces. Using this technique, selected **UCB**s can be made to appear "**above the line**" or "**below the line**" as required in each address space. Shared pages can be used for other purposes as well. Page frames are saved as long as a shared page is not changed. When one is changed, that address space's virtual page is mapped to a different page frame before the update is completed.

simulation model a computer program which attempts to perform like the system or program under study, and which can be monitored or measured to provide planning information. The technique is valuable for estimating the behavior of very large physical or economic systems.

slot unit of auxiliary storage, a 4096-byte physical record on a page data set.

SmartBatch a separately priced add-on to MVS that supersedes the BatchPipes/MVS offering.

The following material is quoted from the from the announcement letter. Note that irrespective of the name, none of SmartBatch requires OS/390. Only the the BatchPipePlex component specifically requires a parallel sysplex.

IBM SmartBatch for OS/390™ is a major enhancement to and a replacement for BatchPipes®/MVS®. It maintains the single system data piping capabilities of BatchPipes/MVS and provides the following enhancements:

- BatchPipePlex™—a new component which extends the data piping capabilities of BatchPipes/MVS to batch jobs operating in a Parallel Sysplex environment. This support satisfies the statement of direction relative to Parallel Sysplex support that was made in Software Announcement 295-400, dated September 26, 1995.
- BATCH ACCELERATOR**—a new component which transparently splits batch jobs into units of work which can be exe-

cuted in parallel on single system images and multiple system images in a sysplex.

- DATA ACCELERATOR** Performance Component—a new component which dynamically restructures I/O requests to improve I/O efficiency for VSAM and non-VSAM access methods.

** Trademark or registered trademark of BMC Software.

SMP-E element of OS/390 (formerly a separate MVS product) used to install and update software in the MVS environment. The availability of SeverPac and other simplified install vehicles in OS/390 does not diminish the need for SMP/E specialists, but their workdays might get shorter.

SMS system-managed storage, the use of hardware and software in MVS/ESA to automate the DASD management or storage management functions usually found in an MVS installation. These functions include space allocation, data set placement, backup, recovery, and archiving or migration of inactive data to less expensive media. SMS in ESA also supports the use of performance considerations in determining data set placement, if only at initial allocation time. The elements of OS/390 that implement SMS are called generically DFSMS, consisting of DFSMSdfp (base functions), DFSMSdss (data mover), DFSMSrmm (removable media manager), and DFSMShsm (archive and backup). Some of these OS/390 elements are optional.

SNA Systems Network Architecture, the network architecture first introduced by IBM in the mid-'70s. SNA is the overall architecture for such elements as VTAM, APPN, SDLC, LU 6.2 and various other networking facilities in the IBM enterprise systems world. It is characterized by many as closed, proprietary, and difficult to maintain—but has supported essential networking for IBM's customers for over two decades. The essential contrast between SNA and **TCP/IP** is that any SNA node must "see" the whole network, out to the ultimate destination. There is no capability to move data point-to-point over numerous alternate paths to the ultimate destination, as in TCP/IP. The idea of a fully determinate network helps SNA to assure delivery and be resistant to penetration. It also makes SNA lack flexibility to deal with rapid growth or change.

SRM System Resources Manager. See Chapter 6 for an extensive discussion.

SRR serially reusable resource.

standard deviation see **normal distribution**

status switch[ing] transition from problem state to supervisor state caused by an interrupt. Status switching is an expensive and disruptive operation because it interrupts the flow of instruction process-

ing, causing the Translation Lookaside Buffer (TLB) and the CPU's High Speed Buffer to be purged either explicitly or implicitly as other virtual storage (and hence central storage) areas are used. Changes in MVS and the hardware it runs on, including the improved PROGRAM CALL facilities in ESA/390 and use of the TEST PENDING INTERRUPT instruction, seek to minimize the need for status switching.

Storage Management Subsystem see **DFSMS** and **SMS**.

storage isolation a facility in compatibility mode that allows selected address spaces to be protected from performance degradation due to page stealing, or to single out address spaces for preferred page stealing. Parameters in the IPS, applied at the performance group period level, define minimum and maximum frame counts or an acceptable page-in rate, or both. See topic A.6 in Appendix A for details and discussion.

storage protection In Workload Manager goal mode, there is no externally specified storage isolation. Instead, the working set behavior of each address space is tracked, and address spaces running behind goal and suffering page-in delay are provided with dynamically adjusted storage protection. (The algorithms are more complex, but this is the general idea.) Storage protection is loosely based on **Working Set Management** as implemented in MVS/ESA SP 4.2, but with the motivation being management of address space performance to goal, rather than optimizing the use of central storage.

SUPERVISOR CALL or SVC the SUPERVISOR CALL instruction, used by a program to request service from the operating system. A code imbedded in the instruction designates the service to be performed; register contents and data addressed through the registers give the details of the request.

swap group in systems without expanded storage, the set of pages that are both swapped out and swapped in. With expanded storage, only the primary working set is swapped in from expanded, and the term does not apply except for direct swaps to auxiliary storage.

swap trim form of page stealing done just prior to swap-out, reducing the frame count of the address space down to the swap group or trimmed working set. Swap trim is aggressive by default; it can be moderated (the swap group expanded) by specifying storage isolation for performance periods (such as first-period TSO) to be protected from page stealing.

synchronous delay delay to a workload that is caused by the activity of the workload itself. For instance, an address space that initiates an I/O operation on a uniprocessor without entering a WAIT state will inevitably be delayed by the back-end processing when the I/O request

is completed. Even with a relatively inactive CPU, the address space will be found in the "waiting for CPU" execution state for a portion of its active time proportional to that kind of activity.

Another example is I/O delay caused when a transaction requires the use of two randomly accessed data sets on the same device or two distinct areas of the same data set.

sysplex as of MVS/ESA SP 4.3, the first level of multi-system integration that preceded parallel sysplex. The major added facilities in 4.3 were XCF and the sysplex timer. With these two additions, true merged console support was possible. A base sysplex (as distinguished from a parallel sysplex) can have a single point of control for console operations, with synchronized clocks on all participating MVS images ensuring properly sequenced time stamps on all messages.

Since SP Version 5, with the coupling facility, sysplex could became parallel sysplex, with data sharing, load sharing and balancing, centralized logging, and global control of workload performance. Parallel sysplex, especially with "cloned" images equally able to receive any work unit, has greatly enhanced manageability, serviceability, and reliability. A parallel sysplex is intrinsically capable of running nonstop for protracted periods of time; IBM is working to remove the last few barriers in OS/390 to true continuous operation.

system-managed storage see **DFSMS** and **SMS**.

T

target working set in compatibility mode, the number of frames held by an address space that will be protected from normal page stealing or migration when storage isolation is specified in its current performance period definition.

TCAS Terminal Control Address Space, an address space invoked during the TSO LOGON process, used to create the address space that will eventually become the TSO session.

TCP/IP a communications protocol (Transmission Control Protocol/Internet Protocol) developed under contract from the U.S. Department of Defense to internetwork dissimilar systems. It is a *de facto* UNIX standard, but is now supported on almost all platforms. TCP/IP is the protocol of the Internet.

The TCP part of TCP/IP provides transport protocol functions which ensure that the total number of bytes sent is received correctly at the destination. The IP part of TCP/IP provides the routing mechanism. TCP/IP is a routable protocol, which means that the messages transmitted contain the address of a destination network as well as a desti-

nation station. This allows TCP/IP messages to be sent to multiple networks within an organization or around the world, hence its use in the worldwide Internet .

Terminal Monitor Program (TMP) the facilitating, service-providing environment that defines a TSO session. TSO was conceived as a generalized interface in which the TMP could be replaced to meet special needs, but in practice, the IBM-supplied TMP is used as is.

tetradic a processor complex configuration in which four processors share storage and a channel subsystem.

text data which consists entirely of coded human-readable characters. Usually, text data is organized as paragraphs of readable data, but source code in a programming language is a special kind of structured text.

think time time during which a user of a terminal-based system is either reading or interpreting the screen content, keying in data without hitting the ENTER key, thinking about the screen and the session, or doing something else entirely. In MVS, it is strictly defined for TSO as the interval between two SYSEVENTs—Terminal Input Wait and User Ready. The recent think time history of an address space is used in the decision process for logical swapping and for swapping to expanded storage.

token passing a multi-node control protocol that uses a special message (the token) that is transmitted to the data link or network by a computer. In one model, there is a single token, constantly circulating. The token can be held by a node while it places data onto the ring. It thereby serves to serialize access to the aggregate resource defined by the ring. Applications of this form of token passing ("passing the buck") within MVS include JES3 and the original form of **GRS.**

In token-ring networks, there are multiple tokens that carry messages or message headers. Token-ring uses multiple tokens emitted by a controlling computer. When a terminal or computer wants to send a message, it waits for an empty token. When it finds one, it fills it with the address of the destination station and some or all of its message. Every computer and terminal on the network constantly monitors the passing tokens to determine if it is a recipient of a message, in which case it "grabs" the message and resets the token status to empty. Token passing uses bus and ring topologies.

token-ring a LAN access method that uses the token passing technology in a physical ring. Each station in the network passes the token on to the station next to it. Token Ring and FDDI LANs use the token ring access method. See **token passing**.

TOR in CICS, Terminal-Owning Region.

TPI TEST PENDING INTERRUPT, an instruction used in MVS/XA and MVS/ESA to allow a processor (CPU) to "pick off" and process a pending I/O interrupt without leaving the I/O interrupt handler. Doing so eliminates a subsequent interrupt and the corresponding disruption to instruction processing caused by status switching. The extent to which TPI is successful is used by the SRM to control the number of processors enabled for I/O interrupts.

transaction-oriented measurement system data collector which captures data reflecting the completion and history of each transaction in a given environment, for later analysis and reporting.

trimmed working set set of frames remaining as part of an address space after swap trim.

TSO Time Sharing Option, the general-purpose interactive method of accessing MVS functions. See chapter 5 for an extended discussion.

tuning performance management activity with the purpose of removing known inefficiency or restoring inadequate service to an acceptable level. In the context of "buy, steal, tune, or accept," to "tune" is to change programs to be more efficient or to change installation choices (such as block sizes) to improve the operation of one workload without penalizing another.

U

UADS User Attribute Data Set, a data set containing identity and control attributes for TSO users.

UCB Unit Control Block, data area in MVS describing the nature and current state of an I/O device. There are as many UCBs as defined devices. When a device is found busy in attempting to start an I/O operation, the queue of requests awaiting the device is associated with the UCB.

UCBBSY bit in the UCB indicating, when set to "1," that an I/O operation is currently active on the device.

UIC Unreferenced Interval Count, measure of activity in central storage. The UIC for a frame is approximately the number of seconds since it was referenced. Lower UICs are usually correlated with higher contention for central storage frames, although it is possible for a workload with a large and stable working set to drive a system's UIC close to zero with little or no contention.

unattended operation see **operator**.

unblocked data set a data set in which the physical record or block is the same size as the logical record. Unblocked data sets with small

block sizes, especially when they are heavily used sequential or partitioned data sets, usually represent a system-wide performance problem.

uncaptured time see **capture ratio**.

underinitiation see **overinitiation**.

Unit Control Block see **UCB**.

UNIX a multiuser, multitasking operating system originally developed by AT&T. UNIX is written in C, also developed by AT&T, which can be compiled into many different machine languages, causing UNIX to run in a wider variety of hardware than any other operating system. For this reason, UNIX has become synonymous with "open systems."

V

V=R attribute of an address space whose virtual addresses are required to match its real addresses. Pages of such an address space cannot be stolen, and the address space cannot be swapped. V=R was used heavily in the infancy of MVS, when it was feared that the burden of I/O address translation would cause incorrect operation of certain critical I/O devices, or that the dynamic address translation process was not to be trusted. V=R has all but disappeared except for some specialized hardware diagnostic programs.

virtual=real see **V=R**.

Virtual Tape a tape subsystem that provides a storage hierarchy for tape volumes. In the IBM implementation, announced for June 1997 delivery, active volumes are virtualized on a large internal RAID DASD cache. Dismounted volumes are maintained in the cache as long as space is available, making quick remounting of the virtual volume possible. When the space is needed, the virtual volume is written to a high-capacity tape volume in an autoloading array, with multiple virtual volumes "stacked" on the real tape up to its full capacity. Inactive virtual volumes are loaded to the DASD cache when they are mounted. Virtual tape makes possible a dramatic reduction in tape inventory.

VLF Virtual Lookaside Facility; in MVS/ESA, used to store named objects such as load modules or TSO CLISTs in data spaces, to save the I/O associated with repeated use.

VM/ESA the current level of IBM's virtual machine operating system.

W

WAIT SVC to suspend the execution of instructions in an address space until an expected event occurs. The usual mechanism is to define

an Event Control Block (ECB), which is the object of the WAIT macro-instruction. Some other process (for instance, another active task in the address space) will later issue the POST macro with the ECB as its object, to signify that the WAIT is completed. Most WAITs are not invoked directly by a problem program but are the implicit consequence of I/O or other service requests.

WAIT state state of a CPU when it has no work to do. The accumulation of WAIT state time is the complement of CPU utilization. Time spent in the WAIT state may not be visible in a logical partition if event-driven scheduling is used.

workload-oriented point of view performance management approach based primarily on the measurement of service delivered by the workloads of interest, with corrective action initiated when a service target is missed. It is the underlying approach of Workload Manager goal mode.

Workplace Shell the graphical user interface of OS/2 Release 3 (Warp).

World Wide Web An Internet service that links documents by providing hypertext links from server to server. It allows a user to jump from document to related document no matter where it is stored on the Internet. World Wide Web client programs, or Web browsers, such as Microsoft Internet Explorer and NetScape Navigator, allow users to browse "the Web."

Developed at the European Center for Nuclear Research (CERN) in Geneva, it was created to link research information between different locations. WWW documents are structured with format codes and hypertext links using the HyperText Markup Language, or HTML. A home page is created for each server with links to other documents locally and throughout the Internet.

The Web has become a centerpiece of Internet activity, because its documents can contain both text and graphics, and it is quickly turning the Internet into an online shopping mall.

work unit the unit of workload management. Work units are batch jobs, TSO, CICS, DDF, IMS, APPC/MVS, or OpenEdition MVS transactions, started tasks, DB2 stored procedures, or enclaves. This enumeration is likely to continue growing.

X

XCF Cross-system Coupling Facility. XCF is the inter-system or inter-image communication facility through which packets of information relating to sysplex operation are exchanged. GRS and JES also

use XCF as a communication medium. The acronym is sometimes rendered as "cross-system communication facility."

Index

Note: This index does not include references to glossary entries.

[Non-Alphabetics]

* in storage isolation 379
3033 Extensions 10
4GL 124
711-based 22
85th percentile
  as "typical" 174

A

ABEND 241
above the line 57, 84
absolute address 53
absolute storage 44
absorption rate 147
access control system 242
access method 53, 93, 95
access register 84
account number 107
Accounting
  See Also Chargeback
accounting data 383
accounting for resource usage 199
ACF2 117-118, 242
active I/O
  delay state 62
  execution state 109
active-time 141
activity rate 197, 207
actuator 219
actuator-level buffering 225
acute contention 214
adaptive resource management 35
address
  absolute 53
address space 3, 8, 16, 77, 83, 369, 383
  attributes 307
  managed 150
  monitored 150
  non-swappable 307
  privileged 309
address spaces 210
addressing
  31-bit 84
addressing mode 84
ADSL 37
AFC
  available frame count 349
AFQ 53, 237-238
  depletion 238
  replenishment 345
ALB 225
alert 203
algorithm
  for MPL control 355
algorithms
  of Workload Manager 162
allegiance 218
ALLOCAS 116
ALLOCATE command
  in TSO 94
allocation xi
allocation recovery 108
alternate path 45
Amdahl 78
Amdahl Corporation 49
analytic model 214
analytic modeling 198
  of I/O 74
APF 264
APG
  See automatic priority group
APGRNG 329
API 200
  See application programming interface
APPC 126
APPC/MVS 35, 55, 101-102, 126

APPC/MVS 19
applet 36
application
  packaged 268
  portable 268
  tuning xv
application architecture 191
application development 26
application packages 269
application performance
  vulnerabilities 268
application program 191, 336
application programmer 269
application programming interface 2
application programs
  backlog 255
application tuning 267–299
applications 256
  design 256
approaches to performance management 198
APU 46
AR
  See access register
architecture 42
  I/O 66
archive 105
array
  natural storage order 276
array storage layout 276
AS/400 127
ASCII 26
ASID 380
ASRV 149, 311
assembly language 272
asymmetric device attachment 45
asynchronous 42
AT&T 24
ATTACH 126
attached processor 46
attributes
  of address space 307
authorized program facility
  See Also APF
automated operations 64
automated tape library 105
automatic priority group 328
auxiliary storage 65, 80, 139, 180, 245, 314, 321, 323, 355, 358, 373
Auxiliary Storage Manager
  ASM 349
  SP2.1.2 changes 15
auxiliary storage shortage 251
  thresholds 251
Auxiliary Storage Shortage Swap 324
auxiliary storage slots
  thresholds 385
availability 207
available frame count 362
available frame queue 144, 237-238, 312, 320
  See also AFQ
available frame queue 80
average response time 172
average seek 220
average seek time 220

B

back-end 63
backing
  of virtual resources 79
backlog
  of application programs 255
backup 105
backup jobs 193
balanced methodology 213
Base Control Program 14
basic design
  flawed 191
batch 1, 32, 40, 100, 102
  non-production 103
batch job 107
  delay factors 107
  performance characteristics 107
Batch Message Processor 116
batch window 193, 263
BatchPipes 36, 114, 188, 190
BCP
  See Base Control Program
Bell Telephone Laboratories 24
Beretvas, Tom xix, 223, 247
BGS Systems 113
bigot
  MVT 2
billing
  incentive for deferred batch 106
binary search 275
binding
  late 94

bipolar 30, 34
block 93
block faulting 346
block multiplexing 68
block multiplexor channel 222
block paging 150
  See Also contiguous slot allocation
block size 74, 95, 144, 259
BMP 116
Boole & Babbage 207, 257
bottleneck 208
bottleneck analysis 210
buffer 54, 76, 95
business growth
  and DP capacity growth 199
business priorities 135
byte 2

C

C (programming language) 24
CA-ACF2 118
cache 70, 220
  DASD controller 15
  store-in 220
  store-through 220
cache control units 70, 257
cache controller 279
cache device 219, 225
caching 264
CADAM 310
CA-LOOK 207
Cambridge Monitor System 79
Candle Corporation 205, 207, 210, 212, 280
  OMEGAMON II 205
canned JCL 105
capacity
  waste
    with under-initiation 111
capacity management 213
capacity planner
  See Capacity Planning
capacity planning xiii
  and performance management 191
capacity projection 213
capping 164
captured storage
  See shared pages
card punch 63
card reader 63, 103
card-image file 190
Carpenter's Fallacy 213
CAT
  See criteria age table
catalog 94
CATALOG 116
catalogued
  data set 94
catalogued procedure 100
CAW
  See Channel Address Word
CBDPO 236
CCCSIGUR 365
  adjustment 365
CCW 57, 67, 93, 218
CCWs 221
CEC 31, 43
central electronic complex 43
central storage 58, 61, 80, 139, 245, 247, 312, 344, 363
  exhaustion 358
  constraint 109
  depletion 239
centralized system
  added value 65
chained searching 275
change 213
  monitoring 212
  ruling out 213
  source of problem 201
  undocumented 213
  unexpected 213
change detection
  limits 213
change management 242
channel 8, 66-67, 218
  cycle-stealing 63
  integrated 63
  operation 67
Channel Address Word 44
channel control word 67
channel measurement block 205
channel program 53, 57, 66
channel set 45, 47
channel set switching 45
Channel Status Word 44
channel subsystem 45, 47
  in XA and ESA 66
channel utilization 207

channels
  association with CPUs 45
  cycle-stealing 45
channel-to-channel adapter 8-9, 43
charge-back 146
Chargeback
  See Also Accounting
checkpoint data set 65
chilled water 28
choices xii
chronic contention 214
CICS xii, 5, 30-31, 40, 50, 55, 82-83, 102, 116, 118, 210, 243, 246, 261, 264, 268, 331, 334
  and APPC/MVS 128
  and paging delay 11
  data tables in hiperspace 246
  internal dispatcher 41
  internal dispatching queue 243
  internal queueing 243
  MRO 243, 261
  response time 189, 197
  Version 4 31
  VSAM buffers in hiperspace 246
  with TSO 11
CICS internal CPU queuing 157
CICS/ESA 21
CICSplex 50
CIO 37
CKD 93
  See count-key-data format
class 102
classification 167
classification group 166
classification rule 166
classification, discretionary 176
client-server 30, 37, 194
  development environment 30
CLIST 254, 367
cloning 53
clustering analysis 308
CMG 244
CMOS 30, 34, 37
  speed of 32, 34
CMS 79
  as Cross-Memory Services 83
  of VM/370 83, 122
CNTCLIST 366
coincident usage 211
command 40
common area 8, 325, 337, 369
common page-in delay 373
Common Service Area
  See Also CSA
common storage 83
common storage area
  See CSA
communications device 63
Compaq 24
compatibility
  release-to-release 84
compatibility mode 207
Compatibility Mode 132-133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155
complexity xiii
composite RV 318
compression, hardware 73
Computer Associates 113, 116-117, 207, 242
concurrent access to resources 192
Concurrent Copy 194
configuration xii
CONN 200
CONNECT 218
connect time 175
connect-time 144
Consent Decree of 1969 28
CONSOLE 116
CONSOLE address space 9
console automation 9
console device 64
constraint 1
contending workload 214
contention 190, 192, 201, 208-209, 243-244, 260, 359
  acute 214
  avoidable 190
  between workloads 103
  chronic 214
  detecting 243
  detection in real-time 262
  inter-system 109
  preventing 243
  prevention 261
  shared DASD 217
  solutions 263
contention analysis 200, 210, 212, 262
contention analyzer 211

contention index 209, 311
  See Also workload
contention,avoidance 261
contention-delay state 261
contiguous slot 111
contiguous slot allocation 249, 321
contiguous-slot allocation 12
continuous operation 53, 194
control mechanisms xiii
control performance group 307, 383
control program xvii
control register 205
control unit 66, 68-69, 218
control unit protocol 219
control units
  cache 70
controlled variable 364
Conversational Monitor System 79
conversion
  to MVS/XA 14
cooperation 298
coprocessor 34
cost of energy 28
cost of swap-in 150, 152
count-key-data 93
  DASD format 70
Couple Data Set 162
coupling facility 50-51
  structure storage 61
Coupling Facility 30, 35
coupling link 52
CP/67 79, 122
CP/CMS 122
CPENABLE 366
  adjustment 366
  and logical partitioning 366
CPGRT 326, 373
CPIC 127
CPU 40
  See Central Processing Unit
  overcommitment 359
  program inefficiency 271
  SDC 370
  service units 142
  well-ordered 120, 137
CPU access 360
CPU configuration 42
CPU delay 355
  from I/O 63
CPU load balancing 365
CPU overcommitment
  desirable 360
CPU queueing 258
CPU queuing delay 207
CPU SDC
  default 370
CPU service
  in goal mode 146
CPU service unit 367
CPU time
  unproductive 150
CPU utilization 207, 359
criteria age 253
Criteria Age Table 337
cross-memory 144
cross-memory addressing 8
cross-memory mode 326
cross-memory services 7-9, 116
  in 3081 10
cross-system ENQ 192
cryptographic 34
CSA 83, 325, 372
CSA allocation
  tracking 203
CSA depletion 203
CSA page data set 248
CSW
  See Channel Status Word
CTC
  See channel-to-channel adapter
CTC ring
  See Also in GRS
customers 240
CWSS 325, 372
cycle time 43
cycle-stealing channels 48

D

DASD xiv, 64, 93
  denial of access 192
  shared 8
  space utilization 278
  string 70
DASD Advisor 257
DASD delay
  in shared environment 253
DASD farm 64

DASD I/O
  modeling 214
DASD management policy 280
DASD modeling 216
DASD tuning
  object of 257
DASDMON 257
DAT
  See Also dynamic address translation
DAT box 3
data
  logical organization 93
data collector 202, 204
Data Control Block 95
data entry time 189
Data Facility Product 14
Data Facility: Storage Management Subsystem 105
Data in Virtual 15, 190
Data Language/I
  See DL/I
data management 2, 94
data processing
  relation to business 198
data set 8, 93
  attribute 94
  moving 63
  partitioned 104
    member 104
  sequential 104
Data Set
  Linear 16
data set control block
  See DSCB
data set growth 256
data set organization 93
data space 16, 84-85, 246, 280
  to enhance integrity 86
  to enhance performance 85
data spaces 190
data streaming 8
data transfer rate 279
data windowing services 255
data-in-virtual 280
Data-in-Virtual 254
dataset
  See data set
DB2 xvi, 30, 102, 116, 118, 280, 334
DCB 95
DCC 68, 222
DCE 195
DD statement
  STEPLIB 126
debugging code 268
default parameters 268
defaults
  dangers of 133
defragmentation 220
degradation analysis 108, 210
delay
  caused by initiator settings 110
  indirect indicators of 355
  on-queue 110
  operator intervention 108
  tape mount 109
delay states 211
delayed 209
demand paging 4
  response time 12
demand paging delay 325
demand paging rate 355
Denning, Peter 273
dependency scheduling
  of batch 105
DEQ 97
  See dequeue
dequeue 8
design
  flawed 191
detected wait 141, 319
device 41
  optical storage 64
device busy 215, 356
device contention 264
device controller 66, 217-218
  contention at 217
device limits
  for paging configuration 356
device order 69
device queueing 216
device response times 356
device rotation 224
device utilization 207, 216
DFHSM 117
DFP
  See Data Facility Product
DFP/370 14

DFSMS 29, 105, 117, 255-256
  data placement decisions 256
direct access storage device
  See DASD
directed VIO 367
directory 94
disabled for interrupts 206, 279
DISC 200
disconnect 68, 218
disconnect time 175
disconnected command chaining 68
Disconnected Command Chaining 222
discretionary 176
disincentives 190
disk
  DASD 64
dispatch 41
dispatcher 41, 53, 205, 320
  SP2.1.7 improvements 15
dispatching priority 3, 7, 64, 108, 118, 126, 139, 162, 179, 209, 306, 328-329, 331, 360, 365, 376
  adjustment by SRM 331
  fixed 329
dispatching queue 330
disruptive address spaces 359
distribution of service 207
DIV 262
  and hiperspaces 255
  See Data in Virtual
DL/I 116
DMN 376
domain 7, 137, 207, 209, 306, 317-318, 359, 369, 374, 376
  assignment to 309
  constraints 209, 309
domain 0 374
domain constraints
  equal 309
donor 161
DOS/VS 3
DP 329, 332, 376
DP budget 199
DPRTY
  JCL parameter 328
  TSO LOGON option 328
DSCB 94-95
DSORG 93
DSRV 149, 311
dual processors 48
dumb terminal 195
duplex page data set 248
DUR 308, 376
DVIO 367
dyadic 31, 47
dynamic address translation 3, 44, 53
  vs. V=R 54
dynamic allocation 95
dynamic caching 255
dynamic channel subsystem 217
dynamic path reconnect 68, 224
dynamic path selection 224
dynamic pathing 224

E

EBCDIC 26
ECC
  See error-correcting code
econometric modeling 267
ECSA 326
effective path busy 224
effectiveness xii
efficiency xii
  volume-dependent 269
elapsed time 187, 213, 262, 318
elements 33
enabled 31
encapsulated 272
End-user
  See Also Customer
energy 27
  cost of 28
engineering design 267
engineering embodiment 42
ENQ 97, 192
  See enqueue
ENQ lockout 98, 310
enqueue 8
  global 8
enqueue delay 210
ENQueue Exchange Swap-out 316
enqueue residence value 367
Enqueue Residence Value 307
ENQueue Swap-in 316
enterprise 103
Enterprise Server Offering 34
Enterprise System Connection 68

Enterprise Systems Architecture 16, 44, 57, 66, 84
  See Also Extended Architecture
EPDM 138, 204
EPILOG 250
EPLPA 84, 325
error recovery 40
error-correcting code 72
ERV 367, 371
  See Enqueue Residence Value
ES/9000 18, 51
ESA 48, 84
  See Enterprise Systems Architecture
ESA/390 18
ESCON 18, 52, 68
ESCRTABX 337
ESCTBDS 253, 338
ESCTVIO 253, 338
Ethernet 26
event-related data 206
exception
  address-translation 77
exception analysis 203
exception message 262
exchange swap 316, 318, 367
exchange swapping 152, 318, 354
EXCP 143
EXEC 254
execution state 108, 200, 281
  page-in delay 109
execution state data 205
execution states 250
  ranking 208
execution-state analysis 208, 210, 213
execution-state data 207, 210, 213, 261
execution-state monitor 211
execution-state profile 262
exit 107
expanded storage 5, 11, 58, 80, 136, 144, 180, 239, 245, 249, 252, 313-314, 320-321, 343, 356, 363, 365, 373, 385
  and UIC 358
  direct swap to 321
  eligibility 307
  in VM 59
  needed by non-swappable work 246
  needed by swappable work 246
  overload 252
  reclaim rate 356
  selection criteria 337
  storage isolation with 325
expanded storage constraint 322
expanded storage controls
  defaults 338
expanded storage criteria 310
experiment 134
expert system 202
exponential distribution 216
exposure 222
extended addressing 84
Extended Architecture 44, 47, 66, 224
extended storage
  as synonym for expanded storage 58
extended swap 10-12, 111, 325
Extended Swap xix
extent 212
externalize 272
externals xi

F

fencing
  See storage isolation
fiber-optic 31
FIFO 372
file 93
file name 94
file system 2
first-come-first-served dispatching 331
fixed dispatching priority 330
fixed frames 367
fixed pages 54
fixed virtual storage 7
fixed-head device 220
floor space 258
flow of control 205
FORTRAN 276
FORTRAN H compiler 275
fragmentation 3
  of data on DASD 62
  real storage, in MVT 2
frame 77
frames
  fixed 367
Friesenborg, Siebo 278
front-end 63

G

Generalized Trace Facility
  See GTF
Gerstner,Louis 36
global enqueue manager
  non-IBM 9
Global Resource Serialization 97
  See GRS
Global SRB 328
goal 161
goal mode 31, 37, 130, 132, 158, 190, 207
  as IPL default 138
goal, velocity 175
Goals, Setting 172
granularity 51
GRS 9, 35, 50, 97, 116, 253
GRS address space 9
GTF 205-206
GUIDE 244

H

happy values 364
hardware xii, 40
  degraded performance 201
hardware compression 73
harvesting 205
hashing algorithm 275
HCD 19
head of string 71
  contention at 217
help desk xii, 29
high-speed buffer 60, 201
Hiperbatch 114
hiperspace 16, 246, 262, 280
  CICS data tables in 246
  direct access to expanded storage 59
  expanded-storage-only) 246
  performance advantage 86
  VSAM buffers in 246
hiperspaces 190, 255
  data windowing services 255
Hitachi Data Systems 78
hold
  batch job 106
HOLD
  JCL parameter 105
  job class 105
horizontal splitting 84, 116
HSM 334
hurricane prediction 202
hypervisor 49, 79

I

I/O
  avoidance 254
  CPU role 66
  in MVS/ESA 62
  physical resource 62
I/O active time 175
I/O addressing
  limitations in System/370 46
I/O analysis 176
I/O architecture 66
I/O completion 62
I/O contention 256, 261
I/O control unit 45
I/O delay 62, 209
I/O device
  delay 210
  use 210
I/O performance degradation
  *vs.*throughput 68
I/O queue 175
I/O queueing
  execution state 109
I/O queueing delay 378
I/O resource 92
I/O subsystem 64
I/O Supervisor 66
I/O tuning 63, 176
I/O WAIT 42
IBM
  2314 222
  Poughkeepsie Laboratory 136
IBM 3380 92
  and paging 12
  models 72
  track format 72
IBM 3880 Model 11 70
IBM 3880 Model 13 70
IBM 3880 Model 21 70
IBM 3880 Model 23 70
IBM 3990 Model 3 70
IBM 4381 48
IBM 709 67
IBM 8100 23
IBM 9121 18
IBM 9221 18

IBM 9340 20
IBM 9672 31, 49, 59
IBM 9674 31
IBM Series 1 23
IBM9021 18
ICS 137, 375, 383
IDAW
  See Indirect Data Address Word
IDMS 116
IEAIPS00 370, 372
IEAOPTxx 136, 336
IEASYS00 137
IEASYSxx 137, 367
Impact Analysis 212, 280
importance 161, 177
IMS xii, 5, 30, 50, 82, 102, 116, 210, 334
  and Virtual Fetch 13
  Control Region 121
  Message Processing Region 121
IMS-DB 118
inactive page 80
incentives and disincentives 190
incident xii
independent software vendor 207
Indirect Data Address Word 57
influence 270
Initialization and Tuning 135, 338, 355, 359, 367, 383, 385
Initialization and Tuning. 251
initiator 104
  active 110
  class assignment 104
  drained 110
  idle 104
  one-class 104
  open 104
initiator settings 111
  modeling 113
initiators 212
  number of 110
input/output
  See I/O
installation xi, 317
Installation Control Specification 383
  See ICS
Installation Performance Specification 369
  See IPS
installation standards 268
in-target 314
integrated channels
  See cycle-stealing channels
Intel 80386 23
Intel Pentium 23
interactive 5
internal contention 280
internal CPU contention 281
internal inefficiency 270
internal reader 103-104
internals xi
Internet 26
inter-processor communication 44
interrupt
  page fault 77
interruption 41
IOC 371
IOP 378
IOQ 372
  =PRTY 372
IOQ=PRTY 378
IOS 66
IOS queue 162
IOS queuing delay 207
IOSQ 175, 200
IOSRVC 370, 373
  IPS parameter 144
IPL 45, 78, 212
IPS xvii, 137, 309, 326, 332, 369-370, 375
  global definitions 370
  required performance groups 375
ISDN 37
ISPF
  and VLF 254
ISV 318

J

Java 36
JCL 94, 100, 270, 375
  DD statement 94
JES 63, 96, 100, 102, 104
  checkpoint data set 65
  reader 104
  SPOOL 65
JES2 15, 30, 43, 96, 102, 115, 334
  CHKPT 236
  job-dependent scheduling 105
  multi-access SPOOL 43
JES3 15, 30, 43, 96, 102, 105, 115
  allocation screening 108, 111
  job-dependent scheduling 105

job 40, 102
  See Also batch
  name 100
  setup 105
JOB card 100
job class 102, 104, 106, 375
  production 107
  scheme 106
job control language 94, 100
  See Also JCL
job entry subsystem 100, 102
Job Entry Subsystem 96
job initiation 102
job management 102
job name 375
job queue 102-103
job selection 102
JOB statement
  PRTY parameter 110
  TIME= parameter 107
jobstream 109

K

K 2
Kina, Gary 244
King, Gary 136, 244
  Workload Characterization 244

L

LAN 24
Landmark 207
  The Monitor for MVS 117
large system 42
  added value 65
late binding 2, 94
latency 221, 223
legend 2, 5, 45
legends 4
Legent 206
Library Lookaside Area
  See LLA
line
  16-megabyte 84
linear data set 254
Linear Data Set 16
linear searching 275
LINK 126
linkage code 272
LINKLIST 125-126
LLA 116, 246, 253
LOAD 126
load balancing 53
load-balancing 111
local area network 24
local page data set 12, 109, 357
locality of reference 55, 274
LOCATE RECORD 221
lock 45, 97
log 205
logging facility 205
logic
  hard-wired 69
logical xiv
  swap controls 362
logical control unit 217
logical partition 78, 246
logical path 217
logical resources 40, 76
logical swap 11, 314, 323, 363
  adjustment 365
  controls 251
  defaults 251
  eligibility 320
  evaluation 363
  grace period 238
logical swap decision 363
  criterion 363
logical swap eligibility 238
logical swapping 56, 238, 260
  defaults 363
  OPT parameters 363
Logical Tuning Approach 160
logically partitioned 212
LOGON procedure 96
long wait 141
LOOK 207
loosely coupled multiprocessing 43
"loved one" 201, 209
low storage 44
LPAR 78, 138, 212, 253, 360
LRU 4-5, 226
LSCTFET 363
LSCTFTT 363
LSCTMTE 363-364
LSCTUCT 363-364
LSCTUCTL 320
LSQA 81, 320
LU 6.2 127

M

M/D/1 queueing model 217
M/M/1 216
  assumptions 216
  exceptions 217
M/M/1 queueing model 216, 224
M0
  default priority 376
main storage 3, 45
mainframe xix, 23, 37
  decline of 23
maintenance
  separation from production 192
managed address space 150
mapping
  of virtual resources to real 79
MASK
  ICS statement 383
Mass Storage Subsystem 78
mathematical model 198
maximum PWSS 381
maxMPL 309-310
MC 205
MCCAECOK 343
MCCAECTH 343
MCCAFCTH 344
MCCASMT1 251, 385
MCCASMT2 251, 385
MCCFXEPR 367
MCCFXTPR 367
MCCMAXSW 349
MDF 49, 212, 246, 253, 360
  domain 246
  real storage with 246
mean 172
mean-time-to-wait 176, 329-330, 365
measurement xii, 27, 138, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235
  real-time 189
measurement systems
  transaction-oriented 205
measurement techniques 205
measurement tool 200
measurement tools 202
  classification 202
megabytes 246
megabytes to MIPS ratio 246
memory 44
  See Also address space
Merrill, Dr. H. W. (Barry) 278
Merrill Consultants 204
MESSAGE
  JCL statement 105
methodology 213
  balanced 213
  workload-oriented 201
MF/1 207
MFT 3
microcode 6, 69, 93
  path selection 218
Microsoft Windows 23
MICS 138, 204
migration 61, 239, 245, 252, 322
migration age 321, 338
  critical 339
millisecond response time 255
Milliseconds Response
  See MSR
minimum resource group 165
minimum seek 220
minimum target working set 380
minMPL 309-310, 316, 374
MIPS 246
MIS
  See Management Information Systems
miscellaneous OPT controls 366
mismanaged workload 213
missed service target 201
mixed workload 11, 243
MLPF 49, 78
modeling 198, 214
  analytic 198
  and capacity planning 214
  of I/O 74
  results 227
  used by Workload Manager 162
modeling DASD I/O 214
modular
  programs 272
monitor xii
  historical 202
  real-time 202
MONITOR CALL 205-206
MONITOR EVENT facility 205
Monitor III 209-210
monitor, performance 98

monitored address space 150
monitoring xii, 193
MP
  See multiprocessor
MP box 44, 46
MP feature 46
MP65 44
MPL 108, 136, 179, 237, 260, 310, 317, 369, 374
  adjustment 354
  delay 175
  See multiprogramming level
  reduction 109
  system-wide 316
MPL adjustment 238, 359
MPL adjustment parameters
  default values 355
MPL control 316
MPL delay 108
MSO
  Service Unit 144
MSR 87
MSS 78
MTTW 365
  algorithm 365
  See mean-time-to-wait
Multiple Domain Facility 78
multiple exposure 222
multiple extents 260
multiple requesting 222
multiple tasks 280
Multiprise 2000 34
multiprocessing 2, 42-43
  in MVT 2
  tightly coupled 43
multiprocessor 42, 243
  side 78
Multiprocessor 43
multiprogramming 2, 42
  in MVT 2
multiprogramming level 136, 237, 314
multiprogramming set 360
multitasking 2, 188
MVM
  early name for MVS 83
MVS xii
  preconfigured offerings 236
  release 14
  under VM 368
  version 14
MVS control blocks 210
MVS nomenclature
  changes 14
MVS System Extensions 6
MVS System Product 7
MVS/370 84, 224, 264
MVS/ESA xvi, 18, 116, 245-246, 251, 253, 256, 264, 280, 372
  full-volume page data set 249
  path delay in 217
MVS/ESA SP 4.2 139
MVS/ESA SP 5.2 33
MVS/ESA SP 5.2.2 33
MVS/ESA SP Version 5 31
MVS/SE
  See MVS System Extensions
MVS/SE1 11, 56, 362, 370
MVS/SP 6
MVS/SP 1.3.0 111
MVS/SP 2.1.7 359
MVS/SP 2.2.0 338
MVS/SP 3.1.0 338
MVS/SP 3.1.3 139
MVS/SP Version 1 84
MVS/SP Version 3
  MVS/ESA 16
MVS/SP2 (SP1.2)
  lack of customers 10
MVS/XA 10, 13
  objectives 14
  path delay in 217
  SP2.3 as last release 16
MVT 2-3, 121
MXG 138, 204

N

negative storage isolation 251, 276, 326, 381
network operating system 24
network-centric computing 36
NIC 158, 268, 352
no-movement seek 220
nonparametric statistics 174
non-production batch 103
non-production jobs 267
nonswappable 337
nonswappable address space 379
nonsynchronous DASD 71, 226

no-op 205
normal distribution 172
nucleus 7-8, 81, 84
numerically intensive 20
numerically intensive computing 268
Numerically Intensive Computing 352

O

OMEGAMON 250, 262
  for CICS 212
OMEGAMON II 205-207, 212
OMEGAMON II for MVS 108, 113, 117
one-class initiator 104
one-dimensional system 243
on-line applications 193
open system 25
OpenEdition 33, 35
open-shop workload 267
operating system 3, 40
Operating System xi
Operating System/360 1
operations xii
operator 103
operator action 212
operator involvement
  in batch management 105
OPT 185, 314, 316, 336-337, 369
  MPL controls 260
OPT parameter 247, 322, 365-366
  ESCTSWWS 321
OPT parameters
  effect of 336
optical storage device 64
optimization
  for performance 270
organizational politics 130
OS
  See Operating System/360
OS/2 24, 127
OS/360 28
OS/360 MP65 44
OS/360 MVT 44
OS/390 1, 24, 33, 35, 37
OS/390 Release 3 175, 200
OS/400 127
OS/VS1 3
oscillation 364
out and ready 141
out-and-ready 316, 318
out-target 314
out-time 141
overcommitment
  of real storage in SVS 4
over-initiation 111
overlap 42
overlapped processing 279
overlay defining 276
override
  of data set attributes 95

P

P/390 236
packaged application 268
page 77
  inactive 80
page data set 5, 11, 65, 80
  local 109
  size considerations 248
page data sets
  number of 249
page fault 5, 217, 312, 325, 380
page fault interrupt 356
page fault rate 136, 251, 355-356, 358
page faults
  in TSO transactions 380
page fixing 54
  for I/O 54
  for performance 54
  long-term 54
page movement 252
  in MVS/370 57
  in MVS/XA and MVS/ESA 58
page movement rate 239, 358
  and CPU delay 358
page stealing 4, 53, 109, 118-119, 139, 162, 237-238, 260, 312-313, 344, 363, 369, 373, 379
  See Also swapping
Page stealing 62
page stealing described 344
pageable link pack area 95, 253
pageable storage shortage 367
PAGEADD 367
page-in 355
page-in delay 109, 355
page-in rate 357, 373
PAGERT2 252

paging 4
  blocked 150
  delay 175
paging configuration 356, 380
  locals-only 12
paging configurations
  with expanded storage 251
paging delay 4, 108, 136, 207, 245, 358
paging devices 356
paging problems 237
paging response time 5, 109
paging subsystem 80, 247, 280
paging-rate control
  of storage isolation 379
Parallel Enterprise Server 32-33, 49
  Generation 3 33
parallel splitting 188
parallel sysplex 31, 35, 37, 49-50, 61, 192
Parallel sysplex 51
Parallel Sysplex
  elements of 51
  Need 50
  Offering 49
Parallel Transaction Server 31, 49
parallel-coupled 31
parameters xii, 42
  system 104
PARMLIB
  See SYS1.PARMLIB
partitioned data set 104
Partitioned Data SetÄExtended 255
Partitioned Data Sets-Extended
  See PDSE
partitioning 48
partitioning, physical 46
partitioning, role in migration 48
path 66
path busy 215, 356
path contention 257
path delay 217
path selection
  microcode 218
path service time 218
path utilization 216, 224
path-busy 218
PC-DOS 79
PCJr 24
PDSE 87, 255
PEND 175, 200, 217
pending 217
pending push-out queue 363
percentile 172, 178
PERFORM 375
performance
  measurement 240
  problems 187, 189, 191, 193, 195
  reporting 240
performance analyst 199
performance degradation 214, 223
  acute 214
performance group 137, 207, 212, 263, 306, 359, 369, 375, 383
  definition 375
  period 137
performance group period 326
performance index 161
performance management xi, 40, 189, 194, 198, 240, 268
  adjustment 135
  approaches 198
  complete methodology 213
  CPU 42
  methodology 213
  proactive 240
  workload view 63
Performance Management 199
  Resource-oriented 199
performance monitor 121, 200, 203, 365
  dispatching priority 121
performance monitoring 178
performance period 307-308, 375
  DUR 308
  parameters 376
performance problem
  console 64
performance problems
  types of 188
Performance Problems
  Preventing 236-237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259, 261, 263, 265
  Solving 236-237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259, 261, 263, 265
performance reporting 241
  unilateral 241
performance target 189
performance targets 240
personal computer 23, 64

PGN 375
physical xiv
physical configuration 336
physical record size 95
physical resources 40
physical sequential 93
physical swap 321
physical swap-out 314
physical swapping 10, 111
PL/1 276
PLPA 84, 95, 125, 253, 260, 325, 372
PLPA page data set 248
politics,organizational 130
portable application 268
positioning 10
POST 118
PPGRT 326, 379, 382
PPGRTR 326, 379, 382
PPT
  See Program Properties Table
PR/SM 17, 49, 78, 212, 246, 253
  real storage with 246
prediction 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235
preemptive dispatching 3
preferred
  for page stealing 326
preferred page stealing 251, 380
prefix register 44-45
prefixing 44, 46
primary working set 320, 322
Principles of Operation 42
printer 63
priority
  selection 102
priority queueing 3
PRIORITY statement 110
private area 8, 77, 80-81
  See Also system area
privileged address space 337, 371, 374
privileged dispatching priority 370
Processor 40
processor address 44
processor complex 31, 43
processor delay 209
Processor Resource/Systems Manager
  See PR/SM
processor storage 64, 80, 139
Processor Storage Estimation 136
processor storage swap 321, 323, 363
processor use 209
production xii
production applications 267
production batch 103
production control 103
production job class 107
productivity
  and response time 123
PROGRAM CALL 117
program check 205
program fetch 125, 253
program object 76
  indirect reference 76
  translation 76
Program Properties Table 309, 371
Program Status Word 41
programmers
  amateur 268
promotion 268, 272
proprietary 25
protected frame count 373
protocol 218
  overlapped 218
  unoverlapped 218
Protocol 218
protocol time 218
prototype 268
PRTY 372
PS/2 24
PSW 44
  See Program Status Word
PTS 32
punched card 72, 102
pushout 313
PVLDP 371
PWSS 326, 379-380
  for TSO 381
  maximum 381

Q

QMF 124
QSAM 93
quality of service 240
Query Management Facility 124
queued I/O
  delay state 62

queued I/O delay 211
queued sequential access method
  See QSAM
queueing
  channels 10
queueing delay 192, 216
queueing model 216
Quickstart service policy 182
quiesce 319

R

R/390 236
RACF 117-118, 242
RAID 25, 32, 194
raised floor 28
RAMAC 32, 35
RAMAC 3 35
RAMAC II 32
randomly accessed data 279
RCCCPUT 359-360
  adjustment 361
  basic recommendation 360
  maximum value 359
RCCFXET 362
RCCFXTT 361
RCCPTRT 238, 247-248, 356
  adjustment 358
  default value 356
  lower limit 357
  upper limit 357
RCCUICT 358
real address 44
real frame replacement
  See page stealing
real storage 4-5, 7, 43, 53, 246, 273, 312
  adequate 360
  constraint 11, 53, 109
  contention 109
  in MVS/ESA 57
  in MVS/XA 57
  management 54
real storage configuration 244
real storage constraint 56, 81, 108, 365
  and 3033MP 56
  consequence of using PR/SM or MDF 60
  in MVS/ESA 60
real storage management 237
  default 237
Real Storage Manager 53, 59, 77, 345
Real Storage Shortage Swap 324
real-time 209
real-time alert 299
real-time monitor 203
real-time monitors 203
reasonableness check 191
reclaim 356
recommendation value 108
recommendations 356
reconfigurability 46
reconnect 68, 219, 223, 226
  dynamic path 68
reenterability 41
reenterable 13
  and other terms 13
reference
  to a page 77
reference bit 53
region 3
register 42
release
  with START I/O 66
Remote Copy 194
Remote Procedure Call 195
renaming
  in MVS 76
replication 263
report class 164, 207
report performance group 383
Request Swap 323
re-reference 279
RESERVE 8, 192
  demotion 9
RESET
  operator command 359, 375
residency 307
resident-time 141
residual path busy 223
Resolve Plus 207
resource
  logical 87
resource accounting 146
resource constraint 136
resource contention 190, 192, 260
resource data 197
resource exhaustion 213, 258
resource group 146, 163-165
resource measurement 206

Resource Measurement Facility
See RMF
resource name
in ENQ 97
resource shortage 201, 208
resource usage
accounting 199
tracking 199
resource-oriented point of view 199
resources xi, 40
resources in contention 212
response time 189, 213, 215, 239, 262
anomalies 239
at terminal 64
vs. RMF transaction completion time 140
Response Time Goals 172
response time target 261, 280
return
from service routine 77
REXX 254
RM1 140, 314
RM2 140, 237, 314, 363
RMF 6, 117, 205-206, 210
Monitor II 250
Monitor III 117, 209-211
Version 5 176
Workload Activity Report 176
RMFGAT 209
RMFWDM 209
RMPTTOM 368
ROSCOE 102
rotation 221
Rotational Position Sensing
See RPS
RPB 223
RPC
See Remote Procedure Call
RPS 223
introduction 223
justification 223
RPS miss 220, 223-225
RSM 279, 345
RTO 378
rule of thumb 200
RV 108

S

SAA 27, 127
SALLOC lock 97
sampling 200, 205, 210
SAS 204
save area 77
scalability 51
scale-up problem 191
scheduled batch 103
scheduling 190, 263
of workloads 193
scheduling preferences 190
scheduling rules 190
scientific applications 267
SCP 6
SDC 370
See service definition coefficient>
SDCs
See service definition coefficients
SEARCH 222, 225
secondary allocation 212
secondary storage 64
upgrade 65
secondary working set 320, 322
second-level interrupt handler 205
sector 221
SEEK 201, 221
segment 81, 85, 320
segment boundary 81
segment table 85
selectable unit
See SU
selection priority 102, 110
self-contention 280
sequential data set 93, 104
sequential data sets 278
sequential delay 221
serially reusable resource 97
server 118
server address space 180
ServerPac 236
service
and service units 142
denying by swapping 359
service agreement 164
service analysis 214
real-time 214
service class 161, 164-165, 207
Service Definition 163
service definition coefficient 146
service definition coefficients 163, 370
in SMF 146

Service Definition Coefficients 370
service exception 213
service expectations 136, 189
  for batch 106
service goal 188
service goals 208
service level agreement 64, 386
Service Level Agreement 130
service level agreements 161
Service Level Reporter 204
service level target 136
service measurement 202
service policy 161
Service Policy 163
service rate 147-148, 163, 318
Service Request Block
  SRB 83
service routine 77
service target 189-190, 194, 201-202
  attainment 189
  missed 201
service targets xii, 138, 241, 258
  de facto 241
service time 189, 215
  path 218
service unit 142
services 40
SET PREFIX 44
SET SECTOR 221
Setting Goals 172
SHARE 244
shared buffering 264
shared DASD 8, 43, 192, 216, 253, 310
  considerations 253
  coupling 253
shared DASD contention 193, 217
shared DASD delay 217
  See Also with GRS
shared libraries
  contention due to 244
shared pages 80, 190
shared real storage 43
shoulder tap 44
side 48
SIGNAL PROCESSOR 44
SIGP 44
single image 46
SIO 66
slot 80
SLR 138, 204
SmartBatch 36, 188, 190
SMB
  See speed matching buffer
SMF 146, 204
SMF exit
  IEFUSI 86
SMP-E 29
SMS 29, 86, 105, 117
SNA 24, 26, 127
Software Configuration 212
software engineering 8
solid-state device 11, 220, 225, 249, 280
sorting algorithms 191
source/sink device 63
SP 4.1 19
SP 4.2.2 21
SP1 7
SP1.3 14
SP2 8
SP2.1.0 14
SP2.1.1 15
SP2.1.2 15
SP2.1.3 15
SP2.1.5 15
SP2.1.7 15
SP2.2 15
SP2.3 16
SP3 (SP1.3) 10
space 27
space utilization 278
sparse reference pattern 280
speed matching buffer 8, 69
SPOOL 65, 102, 236
SQA 372
SRB 83, 370
  for I/O completion 360
  Global 328
  service units 142
SRB time 206
SRM 108, 111, 237, 279, 369
  complexity 139
  concepts 139
  concepts and terminology 132
  evolution 139
  experiments 134
  measurement sampling interval 140
  measurement summarization interval 140
  measures of time in 140

objective 132
overhead 310
service rate 132
terms and concepts 140
SRM 20
SRM constants 385
SRM parameter 185, 336
SRM second 140-141, 371
SRM summarization interval 360
SRR 97
SSD 249, 280
See solid-state device
standard deviation 172
standards
installation 268
START
operator-command 101
START I/O 66-67
START I/O FAST 66-67
START SUBCHANNEL 217
started task 100, 115, 121, 384
states
execution and delay 161
status switch 205
STEPLIB 260, 268, 386
STEPLIB DD statement 126
STIMER 141
stochastic modeling 198
storage
central xiv
primary xiv
secondary xiv
system-managed 65
storage class 87
storage constraint 180, 317, 356, 370, 373, 380
storage controller 69
storage delay 209
storage director 69
storage estimate 5
storage fencing
See Also storage isolation
storage isolation 4, 7, 12, 56, 119, 162, 209, 248, 250, 306, 324-325, 373, 379, 381
and locals-only 12
for TSO 327
negative 251
parameters 325
TSO 250
unfavored address spaces 251
with expanded storage 327
Storage isolation
vs. page fixing 55
storage load balancing 152
storage management 65, 256
storage management subsystem 117
Storage Management Subsystem 86, 105
storage occupancy 279
storage protection 162, 180
Storage Technology Corporation 225
STORE PREFIX 45
string controller 70
string switching 45, 224
SU 7
sub-second response time 130
SUBSYS 383
ICS statement 383
subsystem xv, 5, 40, 55, 101, 167
Subsystem Storage Protection 21, 128
subsystems
interactions among 116
SUPERVISOR CALL 41
SUPERZAP xv
suspend/resume 10
SVC
See SUPERVISOR CALL
SVS 3
Swap Analysis 315, 317-318, 323
Implications 316
swap data set 11, 80, 321-322
swap data sets 249
swap delay 323
tuning of 323
swap group 250, 325
swap packet 348
swap paging 10
swap paging delay 325
swap set 250, 380
swap trim 5, 319, 338, 380
swap working set 338
swap-in 312, 315, 360, 380
initiated by SRM 315
swap-in delay 260, 316
swap-out 314, 380
swappable workload 137, 374
swapping 5, 162
delay 175

of inactive address spaces 319
types of 317
Swapping 61
swapping delay 207
SWAPRSF 152, 351
synchronous delay 210, 217
SYS1.PARMLIB xii, 136–137, 162, 336
SYSEVENTs 140
SYSIN 102
SYSOUT 102
sysplex
parallel 31
Sysplex 49
sysplex 19
Sysplex Timer 36
SYSSTC service class 168, 175
system address space 84
system architecture 14
system crash
from CSA depletion 203
system goal 170
system management 29
System Management Facility
See SMF
system multiprogramming level 108
system operator 103
system parameters 104
system performance 336
system programmer xii, 29, 199
system resources
non-specific 201
System Resources Manager
See SRM
SYSTEM service class 168, 175
system software
licensing 6
system task
See started task
System/360 2, 6, 42, 66, 93, 275
System/360 Model 65MP 44
System/360 Model 91 42
System/370 2-3, 44-45, 66
architectural limitations 14
System/370 Extended Architecture 13, 57, 84
System/370 Multiprocessors 45
system-level tuning 259
system-managed storage 65, 253
Systems management
complexity 27
Systems Network Architecture 127
SystemView 27
system-wide high UIC 237-238, 320-321, 338

T

table organization 275
tape 64
magnetic 105
target 373
target MPL 318
target working set 379
TCAM 122
TCAS 121
TCB 370
TCB mode 142
TCB time 206
TCMP
See Also tightly coupled multiprocessing
TCP/IP 26
technical support xii
technology 43
terminal 63
Terminal Monitor Program 319
terminal wait 321
TERMINAL WAIT 319
SYSEVENT 319
TEST PENDING INTERRUPT 66, 366
text 76
The Monitor 207
The Monitor for MVS 117
think time 124, 189, 338, 363
thrashing 110
throughput 43, 133, 336
tightly coupled multiprocessing 43
time sharing 3
in MVT 2
Time Sharing Option 121
See TSO
time slicing 331
time unit 140, 142, 370
time-line plot 256
time-slice
pattern 370
time-slice group 382
TMON 117, 207
token-ring 26

token-ring network
GRS as 9
Top Secret 117, 242
total service units 165
TPI 66, 366
tracking resource usage 199
transaction 40
transaction-oriented measurement systems 205
Transition Swap 323
triadic 47
trimmed working set 363
trivial response time
variable 125
true-ready queue 320
TSDP 332, 382
TSGRP 382
TSO xiv, 32, 35, 40, 95-96, 101, 121, 126, 243, 254, 267, 314, 334
CALL command 141
CLIST 366
LOGON command 375
LOGON procedure 126
paging delay 380
re-reference page faults 250
response time 11, 332, 362
response time data 206
session 101
stall condition 310
storage isolation 250
swapping 139
transaction 123
transaction, trivial 123
trivial transactions 309
user-ID 101
TSO Command Language 96
TSO page faults 250
TSO response time 206
subsecond 12
TSO swapping
in SP1.3 11
in SVS 4
TSOMON 206
TSPTRN 332, 371
tuning 27, 108, 157
tuning for efficiency 259
tuning opportunities 208
TUNIT 332, 371
two-channel switch 45

U

UCB 216
UCBBSY 216
UIC 54, 207, 338, 355, 358, 363
$Isystem-wide high 237-238
unattended operation 9
unblocked data set 190
unbundling 6
under-initiation 111
unilateral swap 108-109, 317, 358, 367
unilateral swap-in 317
unilateral swap-out 317
unilateral swapping 317
uniprocessor 22, 243
Uniprocessor 42
Unit Control Block
See UCB
unit record device 63
UNIX 24, 27, 33, 35-36
UNIX 95 Profile Brand Certification 35
unproductive CPU time 150
unscheduled batch 103
UNT 376
UNT=R 308
USER name 107
user-id 375
Users
See Also Customers
using 209
utilization 197

V

V=R 54
validity of reference 274
in I/O 279
vector processor 188
vectorization 188
velocity goal 175
vendor application packages 269
vendor offerings 207
Version 4 18
VIO 78, 355
virtual xiv
virtual address 44
Virtual Fetch 13
virtual machine 368
virtual resources 40, 76

virtual storage 3, 7, 41, 53, 95, 268
  inefficiency 273
  layout in MVS/370 81
virtual storage constraint 81, 254, 264
virtual storage constraint relief 8, 82
Virtual Storage Manager 77
Virtual Tape 36
virtual=real
  See V=R
virtualization
  hardware 78
virtualize 77
  to share real resource 78
VLF 246, 254
VM 79, 360
VM/370 122
VM/XA 49
VM/XA System Product 79
volume 269
  See DASD
volume fragmentation 256
volume ownership 269, 280
volume pooling 280
VS
  See virtual storage
VSAM 16, 116, 261
  linear data set 254
VSAM buffers in hiperspace 246
VSCR 82, 84
  See virtual storage constraint relief
VTAM 30, 64, 102, 115, 118, 122

W

WAIT state 41, 117
waiting for ENQueue 209
waiting for I/O 209
warning message 262
Watson & Walker, Inc. 182
Watson, Cheryl 182
well-ordered CPU 120
Windows 95 24
WLM
  See Workload Manager
work area 76
work unit 2, 40, 161, 163
workflow 209
working set 55, 237, 273-274, 319
  defined 273
Working Set Management 160
Working Set Manager 110, 133, 149-150, 358
workload xi, 40, 148, 161, 163, 165, 188
  arbitrary 212
  performance 336
  priorities 137
  scheduling 193
Workload 100-101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131
Workload Characterization 136, 244
workload data 198
workload delay factors 207
Workload Delay Monitor 209
workload exception 262
workload growth 258
workload manager 50
Workload Manager 31, 35, 37, 156, 190, 194, 200
Workload Manager Couple Data Set 163
workload performance problem 208
workload scheduling 263
workload view 194
workload-oriented measurement 208
workload-oriented methodology 201
workload-oriented performance management 200
workload-oriented point of view 194
Workload-oriented tools 299
World Wide Web 26, 37
wrong-way tables 275
WSM
  See Working Set Manager

X

XA 84
  See Extended Architecture
XCF 50, 162
XCTL 126
XMS
  See CMS
XPG4 33

Y

year 2000 26